Chapter 2
VIKOR

2.1 Introduction

According to Opricovic [11], the researcher who originally conceived VIKOR (the acronym is in Serbian: VlseKriterijumska Optimizacija I Kompromisno Resenje, meaning multicriteria optimization and compromise solution), the method has been developed to provide compromise solutions to discrete multiple criteria problems that include non-commensurable and conflicting criteria. It has attracted much attention among researchers and has been applied in various areas (Table 2.1). Its theoretical background is closely related to TOPSIS that was presented in Chapter 1; they are both based on an aggregating function representing the "closeness to the ideal" [15]. VIKOR is considered to be effective in cases where the decision maker cannot be certain how to express his/her preferences coherently and consistently at the initial stages of the system design. Yu [27] and Zeleny [29] provide the setting theory for compromise solutions. Opricovic and Tzeng [15] state that "a compromise solution is a feasible solution, which is closest to the ideal, and a compromise means an agreement established by mutual concessions." As such, the compromise solution can well serve as a basis for negotiations. VIKOR has been successfully applied among other domains to lean tool selection [1], environmental management [2, 3], waste management [10], social sustainability [17], facility location [24], material selection [6], and healthcare management [30]. Like TOPSIS, VIKOR has been combined successfully with many different MCDA methodologies and other techniques. Table 2.1, adopted from [26], presents the distribution of papers on VIKOR by application areas, while Table 2.2, adopted from the same reference, shows techniques that were compared/combined with VIKOR.

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-319-91648-4_2) 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_2

Table 2.1 Distribution of papers on VIKOR by application areas [26]

Area	N	%
Design and manufacturing management	38	19.2
Business and marketing management	35	17.7
Supply chain and logistics management	26	13.1
Environmental resources and energy management	20	10.1
Construction management	9	4.5
Education management	9	4.5
Health care and risk management	7	3.5
Tourism management	7	3.5
Non-application	24	12.1
Other topics	23	11.6
Total	198	100

Table 2.2 Distribution of techniques compared or combined with VIKOR [26]

Techniques compared/combined	N	%	Techniques compared/combined	N	%
Fuzzy approach	89	34.6	Rough-set theory	4	1.6
TOPSIS	30	11.7	BSC	2	0.8
AHP	29	11.3	Genetic algorithm	2	0.8
ANP	27	10.5	Support Vector Machines (SVM)	2	0.8
DEMATEL	17	6.6	WSM	2	0.8
Entropy method	10	3.9	COPRAS	1	0.4
PROMETHEE	9	3.5	QFD	1	0.4
ELECTRE	7	2.7	DEA	1	0.4
Group decision making approach	6	2.3	LINMAP	1	0.4
Delphi method	5	1.9	SERVQUAL	1	0.4
GRA	5	1.9	SWOT	1	0.4
SAW	4	1.6	PCA	1	0.4

2.2 Methodology

The original algorithm, as presented in [15], is composed of five distinct steps and this is the version that we selected to implement here. At a later stage, the method was extended with four additional steps that [12, 16]: (1) provide a stability analysis to determine the weight stability intervals, and (2) include a trade-off analysis. VIKOR has been developed to solve the following problem

$$mco_i \left\{ \left(f_{ij} \left(A_i \right), i = 1, 2, \cdots, m \right), j = 1, 2, \cdots, n \right\} \tag{2.1}$$

where m is the number of feasible alternatives; n is the number of criteria; $A_i = x_1, x_2, \cdots, x_m$ is the ith alternative obtained (generated) with certain values of system variables x; f_{ij} is the value of the jth criterion function for the alternative A_i; and mco denotes the operator of a multicriteria decision making procedure

for selecting the best (compromise) alternative in multicriteria sense. The VIKOR algorithm is comprised of five steps as follows:

Step 1. Determine the Best and the Worst Values of All Criteria Functions
Determine the best f_j^* and the worst f_j^- values of all criteria functions

$$f_j^* = \max_i f_{ij},\ f_j^- = \min_i f_{ij},\ i = 1, 2, \cdots, m,\ j = 1, 2, \cdots, n \tag{2.2}$$

if the jth function is to be maximized (benefit) and

$$f_j^* = \min_i f_{ij},\ f_j^- = \max_i f_{ij},\ i = 1, 2, \cdots, m,\ j = 1, 2, \cdots, n \tag{2.3}$$

if the jth function is to be minimized (cost).

Step 2. Compute the Values S_i and R_i
Compute the values S_i and R_i by the relations

$$S_i = \sum_{j=1}^{n} w_j \left(f_j^* - f_{ij}\right) / \left(f_j^* - f_j^-\right),\ i = 1, 2, \cdots, m,\ j = 1, 2, \cdots, n \tag{2.4}$$

$$R_i = \max_j \left[w_j \left(f_j^* - f_{ij}\right) / \left(f_j^* - f_j^-\right)\right],\ i = 1, 2, \cdots, m,\ j = 1, 2, \cdots, n \tag{2.5}$$

where w_j is the weight of the jth criterion.

Step 3. Compute the Values Q_i
Compute the values Q_i by the relation

$$Q_i = v\left(S_i - S^*\right) / \left(S^- - S^*\right) + (1 - v)\left(R_i - R^*\right) / \left(R^- - R^*\right),\ i = 1, 2, \cdots, m \tag{2.6}$$

where $S^* = \min_i S_i$; $S^- = \max_i S_i$; $R^* = \min_i R_i$; $R^- = \max_i R_i$; and v is introduced as a weight for the strategy of the "majority of criteria" (or the "maximum group utility"), whereas $1 - v$ is the weight of the individual regret. These strategies could be compromised by $v = 0.5$, and here v is modified as $v = (n + 1)/2n$ (derived from $v + 0.5(n - 1)/n = 1$) since the criterion (1 of n) related to R is also included in S.

Step 4. Rank the Alternatives
Rank the alternatives, sorting by the values S, R, and Q in ascending order. The results are three ranking lists.

Step 5. Propose a Compromise Solution
Propose as a compromise solution the alternative $[A^{(1)}]$, which is the best ranked by the measure Q (minimum) if the following two conditions are satisfied:

- C1 - Acceptable advantage

$$Q\left(A^{(2)}\right) - Q\left(A^{(1)}\right) \geq DQ \tag{2.7}$$

  where $A^{(2)}$ is the second ranked alternative by the measure Q and $DQ = 1/(m-1)$.
- C2 - Acceptable stability in decision making: The alternative $A^{(1)}$ must also be the best ranked by S and/or R. This compromise solution is stable within a decision making process, which could be the strategy of maximum group utility ($v > 0.5$), or "by consensus" ($v \approx 0.5$), or "with veto" ($v < 0.5$). If one of the conditions is not satisfied, then a set of compromise solutions is proposed, which consists of:
  - Alternatives $A^{(1)}$ and $A^{(2)}$ if only the condition C2 is not satisfied, or
  - Alternatives $A^{(1)}, A^{(2)}, \cdots, A^{(l)}$ if the condition C1 is not satisfied; $A^{(l)}$ is determined by the relation $Q(A^{(l)}) - Q(A^{(1)}) < DQ$ for maximum l (the positions of these alternatives are "in closeness").

The original version of the VIKOR method concludes in this step; the next four steps are parts of the extended version.

Step 6. Determine the Weight Stability Interval for Each Criterion
Determine the weight stability interval $\left[w_j^L, w_j^U\right]$ for each criterion, separately, with the initial (given) values of weights. The compromise solution obtained with the initial weights (w_j, $j = 1, 2, \cdots, n$) will be replaced at the highest ranked position if the value of a weight is out of the stability interval. The stability interval is only relevant concerning one-dimensional weighting variations.

Step 7. Determine the Trade-Offs
Determine the trade-offs, $tr_{jk} = \left|\left(D_j w_k\right) / \left(D_k w_j\right)\right|, j = 1, 2, \cdots, n, k = 1, 2, \cdots, n, k \neq j$, where tr_{jk} is the number of units of the jth criterion evaluated the same as one unit of the kth criterion, and $D_j = f_j^* - f_j^-, \forall j$. The index j is given by the decision maker.

Step 8. Adjust the Trade-Offs
The decision maker may give a new value of $tr_{jk}, j = 1, 2, \cdots, n, k = 1, 2, \cdots, n, k \neq j$ if he/she does not agree with the computed values. Then, VIKOR performs a new ranking with the new values of weights $w_k = \left(D_k w_j tr_{jk}\right) / D_j$, $j = 1, 2, \cdots, n, k = 1, 2, \cdots, n, k \neq j, w_j = 1$ (or the previous values). VIKOR normalizes the weights, with the sum equal to 1. The trade-offs determined in Step 7 could help the decision maker to assess new values, although that task is very difficult.

Step 9. Algorithm Termination
The VIKOR algorithm ends if new values are not given in Step 8.

The results of VIKOR are rankings by S, R, and Q, the proposed compromise solution (one or a set), the weight stability intervals for a single criterion, and the trade-offs introduced by VIKOR.

2.2.1 Numerical Example

Let's assume that we have to solve the same problem as the one presented in Section 1.2.1; that will offer the opportunity to compare the results with those of TOPSIS. Consequently the data input for the experiment are those of Table 1.2. Initially, we calculate the best f_j^* and worst f_j^- values of all criteria functions using Equation (2.2) (Table 2.3). Next, we calculate the difference of each value in the decision matrix from the ideal values of each criterion (Table 2.4). Then, we calculate the values of S_i and R_i using Equations (2.4) and (2.5) (Table 2.5). Finally, we calculate the values Q_i using Equation (2.6) (Table 2.6). The final results are shown in Table 2.6; the first two rows show the values of S_i and R_i, while the third (scenario (a)) shows the values of Q_i for $v = 0.625$ (in this case $v = (4 + 1)/2 * 4$, since the criteria are four). Scenarios (b)–(d) show the values of Q_i for different values of v. Figure 2.1 displays the results for $v = 0.625$. The best ranked alternative, with good advantage, is site 5 since it has the minimum measure of $Q(0.000)$ and also according to Step 5 of the algorithm:

- C1 - Acceptable advantage: $Q(A^{(4)}) - Q(A^{(5)}) = 0.271$ and $DQ = 1/(m-1) = 1/5 = 0.2$, therefore, $Q(A^{(4)}) - Q(A^{(5)}) \geq DQ$ (condition satisfied).
- C2 - Acceptable stability in decision making: S_5 (0.080) $\leq S_4$ (0.310) and R_5 (0.080) $\leq R_4$ (0.133); hence, site 5 is better ranked by both S and R (condition satisfied).

The results are the same for the scenarios (b)–(d) and interestingly enough, for the same dataset, TOPSIS produces the same best solution, site 5.

If the initial data in Table 1.2 is slightly modified, i.e., the score of site 5 for the first criterion (investment costs) is reduced from 11 to 1 and the score of site 5 for the second criterion (employment needs) is reduced from 10 to 1, then VIKOR yields the results shown in Table 2.7. In this scenario, site 4 has the minimum measure of $Q(0.000)$, but according to Step 5 of the algorithm:

- C1 - Acceptable advantage: $Q(A^{(6)}) - Q(A^{(4)}) = 0.176$ and $DQ = 1/(m-1) = 1/5 = 0.2$, therefore, $Q(A^{(6)}) - Q(A^{(4)}) \ngeq DQ$ (condition not satisfied).
- C2 - Acceptable stability in decision making: S_4 (0.133) $\leq S_6$ (0.290) and R_4 (0.133) $\leq R_6$ (0.150); hence, site 4 is better ranked by both S and R (condition satisfied).

Table 2.3 Best f_j^* and worst f_j^- values of all criteria functions

	Investment costs	Employment needs	Social impact	Environmental impact
Weight	0.4	0.3	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
f_j^*	11	10	7	7
f_j^-	5	3	2	1
$f_j^* - f_j^-$	6	7	5	6

Table 2.4 Difference from the ideal solution

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	3	3	5	6
Site 2	6	7	0	2
Site 3	4	5	1	3
Site 4	2	1	0	4
Site 5	0	0	4	0
Site 6	5	1	2	3

Table 2.5 Calculation of S_i and R_i

	Investment costs	Employment needs	Social impact	Environmental impact	S_i	R_i
Site 1	0.200	0.129	0.100	0.200	0.629	0.200
Site 2	0.400	0.300	0.000	0.067	0.767	0.400
Site 3	0.267	0.214	0.020	0.100	0.601	0.267
Site 4	0.133	0.049	0.000	0.133	0.310	0.133
Site 5	0.000	0.000	0.080	0.000	0.080	0.080
Site 6	0.333	0.049	0.040	0.100	0.516	0.333

Condition C1 is not satisfied and then a set of compromise solutions is proposed consisting of both site 4 and site 6 since only condition 1 is not satisfied (site 4 $\approx$ site 6). In this case, $l = 2$, since $Q(A^{(6)}) - Q(A^{(4)}) = 0.176\ (< DQ)$, $Q(A^{(3)}) - Q(A^{(4)}) = 0.253\ (> DQ)$, and the positions of site 4 and site 6 are "in closeness."

Table 2.6 Final results for the facility location problem using initial data

		Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
	S_i	0.629	0.767	0.601	0.310	0.080	0.516
	R_i	0.200	0.400	0.267	0.133	0.080	0.333
(a)	$Q_i(v = 0.625)$	0.640	1.000	0.693	0.271	0.000	0.693
(b)	$Q_i(v = 0.2)$	0.460	1.000	0.618	0.200	0.000	0.760
(c)	$Q_i(v = 0.5)$	0.587	1.000	0.671	0.250	0.000	0.713
(d)	$Q_i(v = 0.8)$	0.714	1.000	0.724	0.301	0.000	0.667

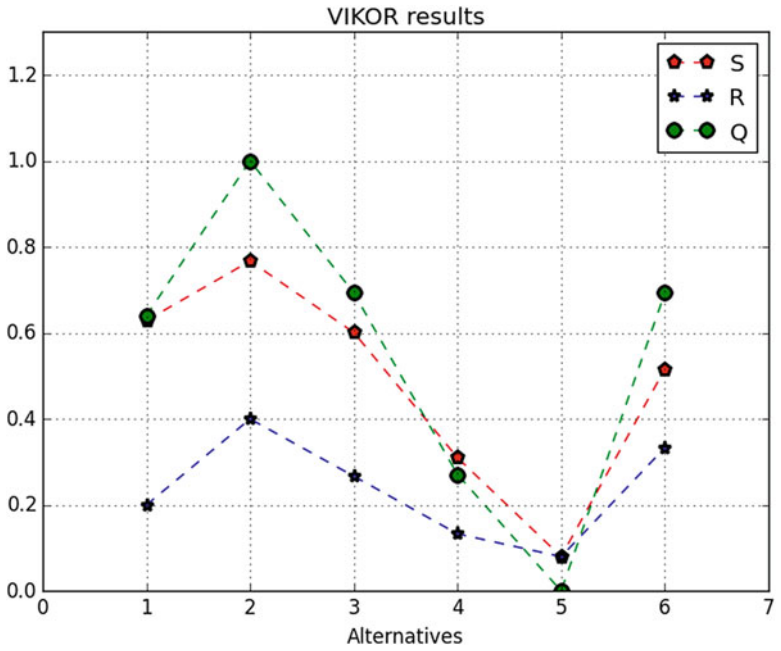

Fig. 2.1 Final results for the facility location problem

Table 2.7 Final results for the facility location problem using modified data

	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
S_i	0.425	0.492	0.370	0.133	0.780	0.290
R_i	0.200	0.225	0.150	0.133	0.400	0.150
$Q_i(v = 0.625)$	0.376	0.476	0.253	0.000	1.000	0.176

2.2.2 Python Implementation

The file *VIKOR.py* includes a Python implementation of the VIKOR method. 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 and output of the function, is embedded as a Python doc string.

More analytically:

- Step 1 (function *best_worst_fij(a, b)*) is in lines 11–23. The inputs are the array with the alternatives performances (variable *a*) and the criteria min/max array (variable *b*). Its output is an array with the best and worst values for all criteria functions (variable *f*).
- Step 2 (function *SR(a, b, c)*) is in lines 26–47. The inputs are the array with the alternatives performances, the array with the best and worst performances, and the criteria min/max array. Its outputs are two vectors with the values of S_i (variable *s*) and R_i (variable *r*).
- Step 3 (function *Q(s, r, n)*) is in lines 50–61. The inputs are the values of S_i and R_i, and the number of criteria. Its output is a vector with the values of Q_i (variable *q*).
- Final results (function *vikor(a, b, c, pl)*) are calculated in lines 64–89. This function calls all the other functions and produces the final result. Variable *a* is the initial decision matrix, *b* is the criteria min/max array, *c* is the criteria weights matrix, and *pl* is whether to plot the results using *matplotlib* or not. Figure 2.1 was produced using the *matplotlib* library.

Using the *timeit* module in lines 102–105 and calling function *vikor* without printing the results (the value of variable *pl* is set to '*n*'), the running time is (an average of 10 runs) 0.0006 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 0.6830 s.

```
1.   # Filename: VIKOR.py
2.   # Description: VIKOR 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: determine the best and worst values for all
10.  # criteria functions
11.  def best_worst_fij(a, b):
12.      """ a is the array with the performances and b is
13.      the criteria min/max array
14.      """
15.      f = zeros((b.shape[0], 2))
```

```
16.      for i in range(b.shape[0]):
17.          if b[i] == 'max':
18.              f[i, 0] = a.max(0)[i]
19.              f[i, 1] = a.min(0)[i]
20.          elif b[i] == 'min':
21.              f[i, 0] = a.min(0)[i]
22.              f[i, 1] = a.max(0)[i]
23.      return f
24.
25.  # Step 2: compute the values S_i and R_i
26.  def SR(a, b, c):
27.      """ a is the array with the performances, b is the
28.      array with the best and worst performances, and
29.      c is the criteria min/max array
30.      """
31.      s = zeros(a.shape[0])
32.      r = zeros(a.shape[0])
33.      for i in range(a.shape[0]):
34.          k = 0
35.          o = 0
36.          for j in range(a.shape[1]):
37.              k = k + c[j] * (b[j, 0] - a[i, j]) \
38.                      / (b[j, 0] - b[j, 1])
39.              u = c[j] * (b[j, 0] - a[i, j]) \
40.                  / (b[j, 0] - b[j, 1])
41.              if u > o:
42.                  o = u
43.                  r[i] = round(o, 3)
44.              else:
45.                  r[i] = round(o, 3)
46.          s[i] = round(k, 3)
47.      return s, r
48.
49.  # Step 3: compute the values Q_i
50.  def Q(s, r, n):
51.      """ s is the vector with the S_i values, r is
52.      the vector with the R_i values, and n is the
53.      number of criteria
54.      """
55.      q = zeros(s.shape[0])
56.      for i in range(s.shape[0]):
57.          q[i] = round((((n + 1) / (2 * n)) *
58.              (s[i] - min(s)) / (max(s) - min(s)) +
59.              (1 - (n + 1) / (2 * n)) *
60.              (r[i] - min(r)) / (max(r) - min(r))), 3)
61.      return q
62.
63.  # VIKOR method: it calls the other functions
64.  def vikor(a, b, c, pl):
65.      """ a is the decision matrix, b is the criteria
66.      min/max array, c is the weights matrix, and pl
67.      is 'y' for plotting the results or any other
68.      string for not
69.      """
```

```
        s, r = SR(a, best_worst_fij(a, b), c)
        q = Q(s, r, len(c))
        if pl == 'y':
            e = [i + 1 for i in range(a.shape[0])]
            plt.plot(e, s, 'p--', color = 'red',
                markeredgewidth = 2, markersize = 8)
            plt.plot(e, r, '*--',  color = 'blue',
                markeredgewidth = 2, markersize=8)
            plt.plot(e, q, 'o--', color = 'green',
                markeredgewidth = 2, markersize = 8)
            plt.legend(['S', 'R', 'Q'])
            plt.xticks(range(a.shape[0] + 2))
            plt.axis([0, a.shape[0] + 1, 0,
                max(maximum(maximum(s, r), q)) + 0.3])
            plt.title("VIKOR results")
            plt.xlabel("Alternatives")
            plt.legend()
            plt.grid(True)
            plt.show()
        return s, r, q

    # 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])

    # criteria max/min
    crit_max_min = array(['max', 'max', 'max', 'max'])

    # final results
    start = timeit.default_timer()
    vikor(x, crit_max_min, w, 'n')
    stop = timeit.default_timer()
    print(stop - start)
    s, r, q = vikor(x, crit_max_min, w, 'y')
    print("S = ", s)
    print("R = ", r)
    print("Q = ", q)
```

2.3 Fuzzy VIKOR for Group Decision Making

2.3.1 Preliminaries: Trapezoidal Fuzzy Numbers

This subsection supplements Section 1.3.1 with additional elements on fuzzy numbers theory to be used for the presentation of fuzzy VIKOR and is based on Lee's work [8]. A trapezoidal fuzzy number A can be defined as $A = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ with a membership function determined as follows (Figure 2.2):

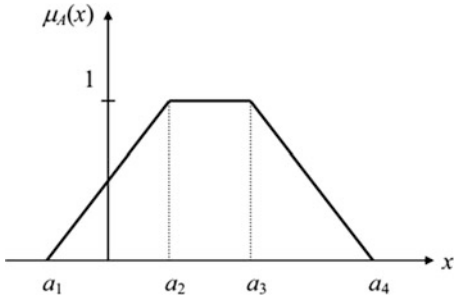

Fig. 2.2 Trapezoidal fuzzy number $A = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, adopted from [8]

$$\mu_A(x) = \begin{cases} 0, & x < \alpha_1 \\ \frac{x-\alpha_1}{\alpha_2-\alpha_1}, & \alpha_1 \le x \le \alpha_2 \\ 1, & \alpha_2 \le x \le \alpha_3 \\ \frac{\alpha_4-x}{\alpha_4}, & \alpha_3 \le x \le \alpha_4 \\ 0, & x > \alpha_4 \end{cases} \tag{2.8}$$

In the case where $\alpha_2 = \alpha_3$, a trapezoidal fuzzy number coincides with a triangular one.

Given a couple of positive trapezoidal fuzzy numbers $A = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ and $B = (b_1, b_2, b_3, b_4)$, the result of the addition and subtraction between trapezoidal fuzzy numbers is also a trapezoidal fuzzy number

$$A(+)B = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)(+)(b_1, b_2, b_3, b_4) = (\alpha_1 + b_1, \alpha_2 + b_2, \alpha_3 + b_3, \alpha_4 + b_4) \tag{2.9}$$

and

$$A(-)B = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)(-)(b_1, b_2, b_3, b_4) = (\alpha_1 - b_1, \alpha_2 - b_2, \alpha_3 - b_3, \alpha_4 - b_4) \tag{2.10}$$

As for multiplication, division, and inverse, the result is not a trapezoidal fuzzy number.

The Center of Area (COA) method, which is used for calculating the crisp value of a fuzzy number in the next subsection, can be expressed using the relation [21]

$$\begin{aligned} \text{defuzz}\left(\widetilde{A}\right) &= \frac{\int x \cdot \mu(x) dx}{\int \mu(x) dx} = \frac{\int_{\alpha_1}^{\alpha_2} \left(\frac{x-\alpha_1}{\alpha_2-\alpha_1}\right) \cdot x dx + \int_{\alpha_2}^{\alpha_3} x dx + \int_{\alpha_3}^{\alpha_4} \left(\frac{\alpha_4-x}{\alpha_4-\alpha_3}\right) \cdot x dx}{\int_{\alpha_1}^{\alpha_2} \left(\frac{x-\alpha_1}{\alpha_2-\alpha_1}\right) dx + \int_{\alpha_2}^{\alpha_3} dx + \int_{\alpha_3}^{\alpha_4} \left(\frac{\alpha_4-x}{\alpha_4-\alpha_3}\right) dx} \\ &= \frac{-\alpha_1\alpha_2 + \alpha_3\alpha_4 + \frac{1}{3}(\alpha_4 - \alpha_3)^2 - \frac{1}{3}(\alpha_2 - \alpha_1)^2}{-\alpha_1 - \alpha_2 + \alpha_3 + \alpha_4} \end{aligned} \tag{2.11}$$

2.3.2 Fuzzy VIKOR Methodology

This section presents a fuzzy extension of VIKOR that is based on the methodology proposed by Sanayei et al. [21]; they use trapezoidal fuzzy numbers and in their paper, they focus on the supplier selection problem but the methodology can easily be applied in a broader scope as well. The fuzzy version of TOPSIS presented in Section 1.3.2 was based on triangular fuzzy numbers; studies of fuzzy VIKOR with this kind of fuzzy numbers have been implemented in the past by Opricovic [13], Rostamzadeh et al. [20], Chen and Wang [4], and Wan et al. [25]. Shemshadi et al. [23], Ju and Wang [7], and Yucenur and Demirel [28] have worked with trapezoidal fuzzy numbers. Other studies in fuzzy VIKOR include Opricovic and Tzeng [14], Park et al. [18], Liu et al. [9], and Devi [5], while Sayadi et al. [22] extended VIKOR with interval numbers.

The steps of the procedure proposed by Sanayei et al. [21] are:

Step 1. Identify the Objectives of the Decision Making Process and Define the Problem Scope
In this step, the decision goals and the scope of the problem are defined. Then, the objectives of the decision making process are identified.

Step 2. Arrange the Decision Making Group and Define and Describe a Finite Set of Relevant Attributes
We form a group of decision makers to identify the criteria and their evaluation scales.

Step 3. Identify the Appropriate 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 4. Pull the Decision Makers' Opinions to Get the Aggregated Fuzzy Weight of Criteria and Aggregated Fuzzy Rating of Alternatives, and Construct a Fuzzy Decision Matrix
Let the fuzzy rating and importance weight of the kth decision maker be $\widetilde{x}_{ijk} = \left(\widetilde{x}_{ijk1}, \widetilde{x}_{ijk2}, \widetilde{x}_{ijk3}, \widetilde{x}_{ijk4}\right)$ and $\widetilde{w}_{ijk} = \left(\widetilde{w}_{ijk1}, \widetilde{w}_{ijk2}, \widetilde{w}_{ijk3}, \widetilde{w}_{ijk4}\right)$, respectively, where $i = 1, 2, \cdots, m$ and $j = 1, 2, \cdots, n$. Hence, the aggregated fuzzy ratings $(\widetilde{x}_{ij})$ of alternatives with respect to each criterion can be calculated as

$$\widetilde{x}_{ij} = \left(x_{ij1}, x_{ij2}, x_{ij3}, x_{ij4}\right) \tag{2.12}$$

where

$$\begin{aligned} x_{ij1} = \min_k \left\{x_{ijk1}\right\}, x_{ij2} = \frac{1}{K}\sum_{k=1}^{K} x_{ijk2}, \\ x_{ij3} = \frac{1}{K}\sum_{k=1}^{K} x_{ijk3}, x_{ij4} = \max_k \left\{x_{ijk4}\right\} \end{aligned} \tag{2.13}$$

The aggregated fuzzy weights ($\widetilde{w}_j$) of each criterion can be calculated as

$$\widetilde{w}_j = \left(w_{j1}, w_{j2}, w_{j3}, w_{j4}\right) \tag{2.14}$$

where

$$\begin{aligned} w_{j1} &= \min_k \left\{w_{jk1}\right\}, w_{j2} = \frac{1}{K}\sum_{k=1}^{K} w_{jk2}, \\ w_{j3} &= \frac{1}{K}\sum_{k=1}^{K} w_{jk3}, w_{ij4} = \max_k \left\{w_{jk4}\right\} \end{aligned} \tag{2.15}$$

The problem can be concisely expressed in matrix format as follows:

$$\widetilde{D} = \begin{bmatrix} \widetilde{x}_{11} & \widetilde{x}_{12} & \cdots & \widetilde{x}_{1n} \\ \widetilde{x}_{21} & \widetilde{x}_{22} & \cdots & \widetilde{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{x}_{m1} & \widetilde{x}_{m2} & \cdots & \widetilde{x}_{mn} \end{bmatrix} \tag{2.16}$$

and the vector of the criteria weights as

$$\widetilde{W} = [\widetilde{w}_1, \widetilde{w}_2, \cdots, \widetilde{w}_n] \tag{2.17}$$

where $\widetilde{x}_{ij}$ and $\widetilde{w}_j, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n$, are linguistic variables according to Step 3. They can be approximated by the trapezoidal fuzzy numbers $\widetilde{x}_{ij} = \left(x_{ij1}, x_{ij2}, x_{ij3}, x_{ij4}\right)$ and $\widetilde{w}_j = \left(w_{j1}, w_{j2}, w_{j3}, w_{j4}\right)$.

Step 5. Defuzzify the Fuzzy Decision Matrix and Fuzzy Weight of Each Criterion into Crisp Values
Deffuzzify the fuzzy decision matrix and fuzzy weight of each criterion into crisp values using COA defuzzification relation (Equation (2.11)). Other methods have also been proposed for this step (for a review, see [19]).

Step 6. Determine the Best and the Worst Values of All Criteria Functions
Determine the best f_j^* and the worst f_j^- values of all criteria functions

$$f_j^* = \max_i f_{ij}, f_j^- = \min_i f_{ij}, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n \tag{2.18}$$

if the jth function is to be maximized (benefit) and

$$f_j^* = \min_i f_{ij}, f_j^- = \max_i f_{ij}, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n \tag{2.19}$$

if the jth function is to be minimized (cost).

Step 7. Compute the Values S_i and R_i
Compute the values S_i and R_i by the relations

$$S_i = \sum_{j=1}^{n} w_j \left(f_j^* - f_{ij}\right) / \left(f_j^* - f_j^-\right), i = 1, 2, \cdots, m, j = 1, 2, \cdots, n \tag{2.20}$$

$$R_i = \max_j \left[w_j \left(f_j^* - f_{ij}\right) / \left(f_j^* - f_j^-\right)\right], i = 1, 2, \cdots, m, j = 1, 2, \cdots, n \tag{2.21}$$

Step 8. Compute the Values Q_i
Compute the values Q_i by the relation

$$Q_i = v\left(S_i - S^*\right) / \left(S^- - S^*\right) + (1 - v)\left(R_i - R^*\right) / \left(R^- - R^*\right), i = 1, 2, \cdots, m \tag{2.22}$$

where $S^* = \min_i S_i$; $S^- = \max_i S_i$; $R^* = \min_i R_i$; $R^- = \max_i R_i$; and v is introduced as a weight for the strategy of the "maximum group utility," whereas $1 - v$ is the weight of the individual regret.

Step 9. Rank the Alternatives
Rank the alternatives, sorting by the values S, R, and Q in ascending order. The results are three ranking lists.

Step 10. Propose a Compromise Solution
Propose as a compromise solution the alternative $[A^{(1)}]$, which is the best ranked by the measure Q (minimum) if the following two conditions are satisfied:

- C1 - Acceptable advantage

$$Q\left(A^{(2)}\right) - Q\left(A^{(1)}\right) \geq DQ \tag{2.23}$$

  where $A^{(2)}$ is the second ranked alternative by the measure Q and $DQ = 1/(m-1)$.
- C2 - Acceptable stability in decision making: The alternative $A^{(1)}$ must also be the best ranked by S and/or R. This compromise solution is stable within a decision making process, which could be the strategy of maximum group utility ($v > 0.5$), or "by consensus" ($v \approx 0.5$), or "with veto" ($v < 0.5$). If one of the conditions is not satisfied, then a set of compromise solutions is proposed, which consists of:
  - Alternatives $A^{(1)}$ and $A^{(2)}$ if only the condition C2 is not satisfied, or
  - Alternatives $A^{(1)}, A^{(2)}, \cdots, A^{(l)}$ if the condition C1 is not satisfied; $A^{(l)}$ is determined by the relation $Q(A^{(l)}) - Q(A^{(1)}) < DQ$ for maximum l (the positions of these alternatives are "in closeness").

2.3.3 Numerical Example

The problem in hand is the same as in the previous examples, the facility location problem. In this case, the importance weights of the qualitative criteria and the ratings are considered as linguistic variables expressed in positive trapezoidal fuzzy numbers, as shown in Tables 2.8 and 2.9; they are also considered to be evaluated by decision makers that are experts on the field. These evaluations are in Tables 2.10 and 2.11.

Initially, we aggregate the weights of criteria to get the aggregated fuzzy weights and the ratings to calculate the fuzzy decision matrix using Equations (2.12)–(2.17) (Table 2.12). Then, we deffuzzify the fuzzy decision matrix and fuzzy weight of each criterion into crisp values using Equation (2.11) (Table 2.13). Next, we calculate the best f_j^* and worst f_j^- values of all criteria functions using Equations (2.18) and (2.19) (Table 2.14). Then, we calculate the values of S_i and R_i using Equations (2.20) and (2.21) (Table 2.15), respectively. Finally, we calculate the values Q_i using Equation (2.22) (Table 2.16). The final results are shown in Table 2.16 and Figure 2.3. The best ranked alternative by the measure Q is site 1, which also satisfies the conditions C1 and C2.

Table 2.8 Linguistic variables for the criteria

Linguistic variables for the importance weight of each criterion	
Very low (VL)	(0, 0, 0.1, 0.2)
Low (L)	(0.1, 0.2, 0.2, 0.3)
Medium low (ML)	(0.2, 0.3, 0.4, 0.5)
Medium (M)	(0.4, 0.5, 0.5, 0.6)
Medium high (MH)	(0.5, 0.6, 0.7, 0.8)
High (H)	(0.7, 0.8, 0.8, 0.9)
Very high (VH)	(0.8, 0.9, 1.0, 1.0)

Table 2.9 Linguistic variables for the ratings

Linguistic variables for the ratings	
Very poor (VP)	(0.0, 0.0, 0.1, 0.2)
Poor (P)	(0.1, 0.2, 0.2, 0.3)
Medium poor (MP)	(0.2, 0.3, 0.4, 0.5)
Fair (F)	(0.4, 0.5, 0.5, 0.6)
Medium good (MG)	(0.5, 0.6, 0.7, 0.8)
Good (G)	(0.7, 0.8, 0.8, 0.9)
Very good (VG)	(0.8, 0.9, 1.0, 1.0)

Table 2.10 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 2.11 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

2.3.4 *Python Implementation*

The file *FuzzyVIKOR.py* includes a Python implementation of the Fuzzy VIKOR method. Similar to the implementation of VIKOR, 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 function, e.g., the input of the function, is embedded as a Python doc string.

More analytically:

- Steps 1–3 of the fuzzy VIKOR procedure are in lines 159–193. Dictionaries *cw* (lines 159–162) and *r* (lines 166–169) correspond to Tables 2.8 and 2.9, respectively; they are the definitions of the linguistic variables used for the criteria weights and the ratings. Python list *cdw* (lines 172–173) is about the importance weight of the criteria (Table 2.10) and the ratings of the six candidate sites by the three decision makers are each in its own list, named *c1*, *c2*, $\cdots$, *c6*

Table 2.12 Fuzzy decision matrix and fuzzy weights

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	(0.500, 0.767, 0.833, 1.000)	(0.400, 0.567, 0.633, 0.800)	(0.100, 0.233, 0.267, 0.500)	(0.700, 0.833, 0.867, 1.000)
Site 2	(0.200, 0.433, 0.467, 0.600)	(0.400, 0.733, 0.767, 1.000)	(0.500, 0.767, 0.833, 1.000)	(0.200, 0.467, 0.533, 0.800)
Site 3	(0.200, 0.467, 0.533, 0.800)	(0.500, 0.700, 0.800, 1.000)	(0.200, 0.433, 0.467, 0.600)	(0.100, 0.233, 0.267, 0.500)
Site 4	(0.500, 0.800, 0.900, 1.000)	(0.700, 0.833, 0.867, 1.000)	(0.500, 0.767, 0.833, 1.000)	(0.000, 0.233, 0.267, 0.600)
Site 5	(0.000, 0.333, 0.367, 0.900)	(0.000, 0.167, 0.233, 0.500)	(0.700, 0.833, 0.867, 1.000)	(0.500, 0.667, 0.733, 0.900)
Site 6	(0.400, 0.700, 0.700, 0.900)	(0.200, 0.467, 0.533, 0.800)	(0.400, 0.667, 0.733, 1.000)	(0.100, 0.333, 0.367, 0.600)
Weight	(0.700, 0.867, 0.933, 1.000)	(0.400, 0.733, 0.767, 1.000)	(0.200, 0.467, 0.533, 0.800)	(0.500, 0.767, 0.833, 1.000)

Table 2.13 Crisp values for decision matrix and weight of each criterion

	Investment costs	Employment needs	Social impact	Environmental impact
Site 1	0.769	0.769	0.789	0.850
Site 2	0.600	0.500	0.850	0.700
Site 3	0.282	0.500	0.769	0.667
Site 4	0.850	0.750	0.282	0.500
Site 5	0.418	0.418	0.415	0.700
Site 6	0.718	0.282	0.231	0.350
Weight	0.870	0.718	0.500	0.769

Table 2.14 Best f_j^* and worst f_j^- values of all criteria functions

	Investment costs	Employment needs	Social impact	Environmental impact
Weight	0.870	0.718	0.500	0.769
Site 1	0.769	0.769	0.789	0.850
Site 2	0.600	0.500	0.850	0.700
Site 3	0.282	0.500	0.769	0.667
Site 4	0.850	0.750	0.282	0.500
Site 5	0.418	0.418	0.415	0.700
Site 6	0.718	0.282	0.231	0.350
f_j^*	0.850	0.769	0.850	0.850
f_j^-	0.282	0.282	0.231	0.350
$f_j^* - f_j^-$	0.568	0.487	0.619	0.500

Table 2.15 Calculation of S_i and R_i

	S_i	R_i
Site 1	0.173	0.124
Site 2	1.010	0.397
Site 3	1.613	0.87
Site 4	1.025	0.538
Site 5	1.761	0.662
Site 6	2.189	0.769

Table 2.16 Final results for the facility location problem

	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
S_i	0.173	1.010	1.613	1.025	1.761	2.189
R_i	0.124	0.397	0.870	0.538	0.662	0.769
$Q_i (v = 0.625)$	0.000	0.397	0.821	0.472	0.763	0.949

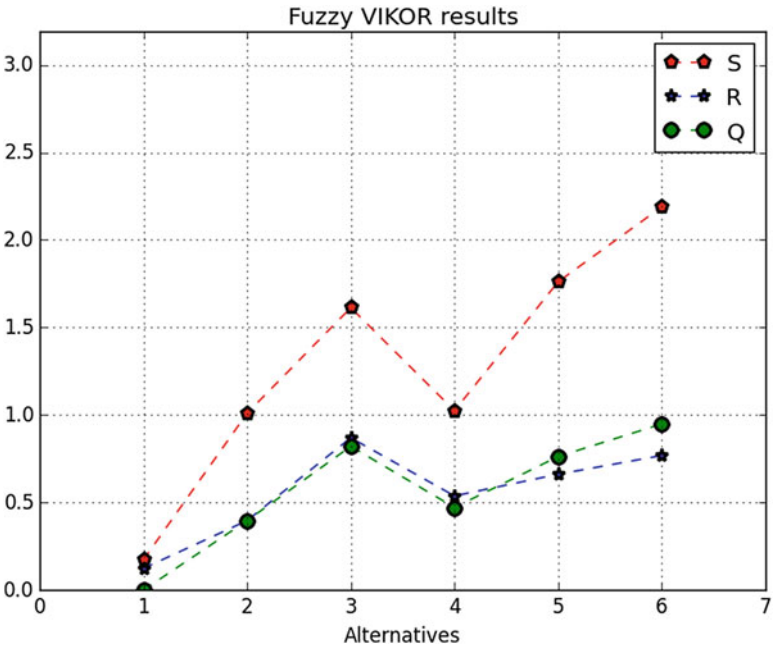

Fig. 2.3 Final results for the facility location problem

(lines 177–188, Table 2.11). These lists are concatenated into one list in line 190. The criteria min/max array is defined in line 193.

- Step 4 is in lines 12–38. Function *agg_fuzzy_value* takes as input a dictionary with the linguistic variables for the importance weight of each criterion or the linguistic variables for the ratings (a), the matrix with the importance weights of the criteria or the ratings by the decision makers (b), and the number of the decision makers (k). Array f is the output of this function and stores the fuzzy decision matrix or fuzzy weights (Table 2.12).
- Step 5 is in lines 42–49. Function *defuzz* takes as input a fuzzy trapezoidal number and returns a crisp value using Equation (2.11) (Table 2.13).
- Step 6 (function *best_worst_fij(a, b)*) is in lines 53–65. The inputs are the array with the alternatives performances (variable a) and the criteria min/max array (variable b). Its output is an array with the best and worst values for all criteria functions (variable f, Table 2.14).
- Step 7 (function *SR(a, b, c)*) is in lines 68–89. The inputs are the array with the alternatives performances, the array with the best and worst performances, and the criteria min/max array. Its outputs are two vectors with the values of S_i (variable s) and R_i (variable r, Table 2.15)
- Step 8 (function *Q(s, r, n)*) is in lines 92–103. The inputs are the values of S_i and R_i, and the number of criteria. Its output is a vector with the values of Q_i (variable q, Table 2.16).

- Final results (function *f_vikor(a, b, c, d, e, n, m, k, pl)*) are calculated in lines 105–150. 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 *e* is the criteria max_min array, 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 2.3 was produced using the *matplotlib* library.

Using the *timeit* module in lines 196–200 and calling function *f_vikor* without printing the results (the value of variable *pl* is set to '*n*'), the running time is (an average of 10 runs) 0.0016 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 0.7164 s.

```
1.   # Filename: FuzzyVIKOR.py
2.   # Description: Fuzzy VIKOR method
3.   # Authors: Papathanasiou, J. & Ploskas, N.
4.
5.   from numpy import *
6.   import matplotlib.pyplot as plt
7.   import timeit
8.
9.   # Step 4: Convert the linguistic variables for the criteria
10.  # weights or the ratings into fuzzy weights and fuzzy
11.  # decision matrix, respectively
12.  def agg_fuzzy_value(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
18.      is the fuzzy decision matrix or the fuzzy
19.      weights of the criteria
20.      """
21.      f = zeros((len(b), 4))
22.      for j in range(len(b)):
23.          k0 = a[b[j][0]][0]
24.          k1 = 0
25.          k2 = 0
26.          k3 = a[b[j][0]][3]
27.          for i in range (len(b[1])):
28.              if k0 > a[b[j][i]][0]:
29.                  k0 = a[b[j][i]][0]
30.              k1 = k1 + a[b[j][i]][1]
31.              k2 = k2 + a[b[j][i]][2]
32.              if k3 < a[b[j][i]][3]:
33.                  k3 = a[b[j][i]][3]
34.          f[j][0] = round(k0, 3)
```

```
            f[j][1] = round(k1 / k, 3)
            f[j][2] = round(k2 / k, 3)
            f[j][3] = round(k3, 3)
        return f

    # Step 5: Deffuzify a trapezoidal fuzzy number into
    # a crisp value
    def defuzz(a):
        """ a is a trapezoidal matrix. The output is a
        crisp value
        """
        return (-a[0] * a[1] + a[2] * a[3] +
            1 / 3 * (a[3] - a[2])**2 -
            1 / 3 * (a[1] - a[0])**2) \
            / (-a[0] - a[1] + a[2] + a[3])

    # Step 6: Determine the best and worst values for all
    # criteria functions
    def best_worst_fij(a, b):
        """ a is the array with the performances and b is
        the criteria min/max array
        """
        f = zeros((b.shape[0], 2))
        for i in range(b.shape[0]):
            if b[i] == 'max':
                f[i, 0] = a.max(0)[i]
                f[i, 1] = a.min(0)[i]
            elif b[i] == 'min':
                f[i, 0] = a.min(0)[i]
                f[i, 1] = a.max(0)[i]
        return f

    # Step 7: Compute the values S_i and R_i
    def SR(a, b, c):
        """ a is the array with the performances, b is the
        array with the best and worst performances, and c
        is the criteria min/max array
        """
        s = zeros(a.shape[0])
        r = zeros(a.shape[0])
        for i in range(a.shape[0]):
            k = 0
            o = 0
            for j in range(a.shape[1]):
                k = k + c[j] * (b[j, 0] - a[i, j]) \
                        / (b[j, 0] - b[j, 1])
                u = c[j] * (b[j, 0] - a[i, j]) \
                    / (b[j, 0] - b[j, 1])
                if u > o:
                    o = u
                    r[i] = round(o, 3)
                else:
                    r[i] = round(o, 3)
            s[i] = round(k, 3)
```

```
89.      return s, r
90.
91.  # Step 8: compute the values Q_i
92.  def Q(s, r, n):
93.      """ s is the vector with the S_i values, r is
94.      the vector with the R_i values, and n is the
95.      number of criteria
96.      """
97.      q = zeros(s.shape[0])
98.      for i in range(s.shape[0]):
99.          q[i] = round((((n + 1) / (2 * n)) *
100.             (s[i] - min(s)) / (max(s) - min(s)) +
101.             (1 - (n + 1) / (2 * n)) *
102.             (r[i] - min(r)) / (max(r) - min(r))), 3)
103.     return q
104.
105. def f_vikor(a, b, c, d, e, n, m, k, pl):
106.     """ a is the dictionary with the linguistic variables
107.     for the criteria weights, b is the matrix with the
108.     importance weights of the criteria, c is a
109.     dictionary with the linguistic variables for the
110.     ratings, d is the matrix with all the ratings, e
111.     is the criteria max_min array, n is the number
112.     of criteria, m is the number of the alternatives,
113.     k is the number of the decision makers, and pl
114.     is 'y' for plotting the results
115.     """
116.
117.     w = agg_fuzzy_value(a, b, k)
118.     f_rdm_all = agg_fuzzy_value(c, d, k)
119.     crisp_weights = zeros(n)
120.     for i in range(n):
121.         crisp_weights[i] = round(defuzz(w[i]), 3)
122.     crisp_alternative_ratings = zeros((m, n))
123.     k = 0
124.     for i in range(n):
125.         for j in range(m):
126.             crisp_alternative_ratings[j][i] = \
127.                 round(defuzz(f_rdm_all[k]), 3)
128.             k = k + 1
129.     s, r = SR(crisp_alternative_ratings,
130.         best_worst_fij(crisp_alternative_ratings, e),
131.         crisp_weights)
132.     q = Q(s, r, len(w))
133.     if pl == 'y':
134.         h = [i + 1 for i in range(m)]
135.         plt.plot(h, s, 'p--', color = 'red',
136.            markeredgewidth = 2, markersize=8)
137.         plt.plot(h, r, '*--',  color = 'blue',
138.            markeredgewidth = 2, markersize = 8)
139.         plt.plot(h, q, 'o--', color = 'green',
140.            markeredgewidth = 2, markersize = 8)
141.         plt.legend(['S', 'R', 'Q'])
142.         plt.xticks(range(m + 2))
```

```
143.         plt.axis([0, m + 1, 0,
144.            max(maximum(maximum(s, r), q)) + 1])
145.         plt.title("Fuzzy VIKOR results")
146.         plt.xlabel("Alternatives")
147.         plt.legend()
148.         plt.grid(True)
149.         plt.show()
150.     return s, r, q
151.
152. m = 6 # the number of the alternatives
153. n = 4 # the number of the criteria
154. k = 3 # the number of the decision makers
155.
156. # Steps 1, 2 and 3
157. # Define a dictionary with the linguistic variables for the
158. # criteria weights
159. cw = {'VL':[0, 0, 0.1, 0.2], 'L':[0.1, 0.2, 0.2, 0.3],
160.     'ML':[0.2, 0.3, 0.4, 0.5], 'M':[0.4, 0.5, 0.5, 0.6],
161.     'MH':[0.5, 0.6, 0.7, 0.8], 'H':[0.7, 0.8, 0.8, 0.9],
162.     'VH':[0.8, 0.9, 1, 1]}
163.
164. # Define a dictionary with the linguistic variables for the
165. # ratings
166. r = {'VP':[0.0, 0.0, 0.1, 0.2], 'P':[0.1, 0.2, 0.2, 0.3],
167.     'MP':[ 0.2, 0.3, 0.4, 0.5], 'F':[0.4, 0.5, 0.5, 0.6],
168.     'MG':[0.5, 0.6, 0.7, 0.8], 'G':[0.7, 0.8, 0.8, 0.9],
169.     'VG':[0.8, 0.9, 1.0, 1.0]}
170.
171. # The matrix with the criteria weights
172. cdw = [['H', 'VH', 'VH'], ['M', 'H', 'VH'],
173.    ['M', 'MH', 'ML'], ['H', 'VH', 'MH']]
174.
175. # The ratings of the six candidate sites by the decision
176. # makers under all criteria
177. c1 = [['VG', 'G', 'MG'], ['F', 'MG', 'MG'],
178.    ['P', 'P', 'MP'], ['G', 'VG', 'G']]
179. c2 = [['MP', 'F', 'F'], ['F', 'VG', 'G'],
180.    ['MG', 'VG', 'G'], ['MG', 'F', 'MP']]
181. c3 = [['MG', 'MP', 'F'], ['MG', 'MG', 'VG'],
182.    ['MP', 'F', 'F'], ['MP', 'P', 'P']]
183. c4 = [['MG', 'VG', 'VG'], ['G', 'G', 'VG'],
184.    ['MG', 'VG', 'G'], ['VP', 'F', 'P']]
185. c5 = [['VP', 'P', 'G'], ['P', 'VP', 'MP'],
186.    ['G', 'G', 'VG'], ['G', 'MG', 'MG']]
187. c6 = [['F', 'G', 'G'], ['F', 'MP', 'MG'],
188.    ['VG', 'MG', 'F'], ['P', 'MP', 'F']]
189.
190. all_ratings = vstack((c1, c2, c3, c4, c5, c6))
191.
192. # criteria max/min array
193. crit_max_min = array(['max', 'max', 'max', 'max'])
194.
195. # final results
196. start = timeit.default_timer()
```

```
197. f_vikor(cw, cdw, r, all_ratings, crit_max_min, n, m,
198.    k, 'n')
199. stop = timeit.default_timer()
200. print(stop - start)
201. s, r, q = f_vikor(cw, cdw, r, all_ratings,
202.    crit_max_min, n, m, k, 'y')
203. print("S = ", s)
204. print("R = ", r)
205. print("Q = ", q)
```

References

1. Anvari, A., Zulkifli, N., & Arghish, O. (2014). Application of a modified VIKOR method for decision-making problems in lean tool selection. *The International Journal of Advanced Manufacturing Technology, 71*(5–8), 829–841.
2. Chang, C. L., & Hsu, C. H. (2009). Multi-criteria analysis via the VIKOR method for prioritizing land-use restraint strategies in the Tseng-Wen reservoir watershed. *Journal of Environmental Management, 90*(11), 3226–3230.
3. Chang, C. L., & Hsu, C. H. (2011). Applying a modified VIKOR method to classify land subdivisions according to watershed vulnerability. *Water Resources Management, 25*(1), 301–309.
4. Chen, L. Y., & Wang, T. C. (2009). Optimizing partners' choice in IS/IT outsourcing projects: The strategic decision of fuzzy VIKOR. *International Journal of Production Economics, 120*(1), 233–242.
5. Devi, K. (2011). Extension of VIKOR method in intuitionistic fuzzy environment for robot selection. *Expert Systems with Applications, 38*(11), 14163–14168.
6. Jahan, A., Mustapha, F., Ismail, M. Y., Sapuan, S. M., & Bahraminasab, M. (2011). A comprehensive VIKOR method for material selection. *Materials & Design, 32*(3), 1215–1221.
7. Ju, Y., & Wang, A. (2013). Extension of VIKOR method for multi-criteria group decision making problem with linguistic information. *Applied Mathematical Modelling, 37*(5), 3112–3125.
8. Lee, K. H. (2006). *First course on fuzzy theory and applications* (Vol. 27). Berlin: Springer Science & Business Media.
9. Liu, H. C., Liu, L., Liu, N., & Mao, L. X. (2012). Risk evaluation in failure mode and effects analysis with extended VIKOR method under fuzzy environment. *Expert Systems with Applications, 39*(17), 12926–12934.
10. Liu, H. C., You, J. X., Fan, X. J., & Chen, Y. Z. (2014). Site selection in waste management by the VIKOR method using linguistic assessment. *Applied Soft Computing, 21*, 453–461.
11. Opricovic, S. (1998). *Multicriteria optimization of civil engineering systems*. PhD Thesis, Faculty of Civil Engineering, Belgrade.
12. Opricovic, S. (2009). A compromise solution in water resources planning. *Water Resources Management, 23*(8), 1549–1561.
13. Opricovic, S. (2011). Fuzzy VIKOR with an application to water resources planning. *Expert Systems with Applications, 38*(10), 12983–12990.
14. Opricovic, S., & Tzeng, G. H. (2002). Multicriteria planning of post-earthquake sustainable reconstruction. *Computer-Aided Civil and Infrastructure Engineering, 17*(3), 211–220.
15. Opricovic, S., & Tzeng, G. H. (2004). Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. *European Journal of Operational Research, 156*(2), 445–455.
16. Opricovic, S., & Tzeng, G. H. (2007). Extended VIKOR method in comparison with outranking methods. *European Journal of Operational Research, 178*(2), 514–529.

17. Papathanasiou, J., Ploskas, N., Bournaris, T., & Manos, B. (2016). A decision support system for multiple criteria alternative ranking using TOPSIS and VIKOR: A case study on social sustainability in agriculture. In S. Liu et al. (Eds.), *Decision support systems vi – decision support systems addressing sustainability & societal challenges. Lecture notes in business information processing* (Vol. 250, pp. 3–15). New York: Springer.
18. Park, J. H., Cho, H. J., & Kwun, Y. C. (2011). Extension of the VIKOR method for group decision making with interval-valued intuitionistic fuzzy information. *Fuzzy Optimization and Decision Making, 10*(3), 233–253.
19. Ploskas, N., Papathanasiou, J., & Tsaples, G. (2017). Implementation of an extended fuzzy VIKOR method based on triangular and trapezoidal fuzzy linguistic variables and alternative deffuzification techniques. In I. Linden et al. (Eds.), *Decision support systems vii – data, information and knowledge in decision support systems. Lecture notes in business information processing*. New York: Springer.
20. Rostamzadeh, R., Govindan, K., Esmaeili, A., & Sabaghi, M. (2015). Application of fuzzy VIKOR for evaluation of green supply chain management practices. *Ecological Indicators, 49*, 188–203.
21. Sanayei, A., Mousavi, S. F., & Yazdankhah, A. (2010). Group decision making process for supplier selection with VIKOR under fuzzy environment. *Expert Systems with Applications, 37*(1), 24–30.
22. Sayadi, M. K., Heydari, M., & Shahanaghi, K. (2009). Extension of VIKOR method for decision making problem with interval numbers. *Applied Mathematical Modelling, 33*(5), 2257–2262.
23. Shemshadi, A., Shirazi, H., Toreihi, M., & Tarokh, M. J. (2011). A fuzzy VIKOR method for supplier selection based on entropy measure for objective weighting. *Expert Systems with Applications, 38*(10), 12160–12167.
24. Tzeng, G. H., Teng, M. H., Chen, J. J., & Opricovic, S. (2002). Multicriteria selection for a restaurant location in Taipei. *International Journal of Hospitality Management, 21*(2), 171–187.
25. Wan, S. P., Wang, Q. Y., & Dong, J. Y. (2013). The extended VIKOR method for multi-attribute group decision making with triangular intuitionistic fuzzy numbers. *Knowledge-Based Systems, 52*, 65–77.
26. Yazdani, M., & Graeml, F. R. (2014). VIKOR and its applications: A state-of-the-art survey. *International Journal of Strategic Decision Sciences, 5*(2), 56–83.
27. Yu, P. L. (1973). A class of solutions for group decision problems. *Management Science, 19*(8), 936–946.
28. Yucenur, G. N., & Demirel, N. Ç. (2012). Group decision making process for insurance company selection problem with extended VIKOR method under fuzzy environment. *Expert Systems with Applications, 39*(3), 3702–3707.
29. Zeleny, M. (1982). *Multi criteria decision making*. New York: McGraw-Hills.
30. Zeng, Q. L., Li, D. D., & Yang, Y. B. (2013). VIKOR method with enhanced accuracy for multiple criteria decision making in healthcare management. *Journal of Medical Systems, 37*(2), 1–9.