MethodsX 8 (2021) 101344

Contents lists available at ScienceDirect

MethodsX

journal homepage: www.elsevier.com/locate/mex

Method Article

Implementation of the new easy approach to fuzzy multi-criteria decision aid in the field of management

Paweł Ziemba

Faculty of Economics, Finance and Management, University of Szczecin, Cukrowa 8, Szczecin 71-004, Poland

A B S T R A C T

Decision-making is one of the most important management functions and a critical task for managers. The tools that support decision makers in making decisions are Multi-criteria Decision Making/Aid/Analysis (MCDM/MCDA) methods. Since most decisions are made under conditions of uncertainty, the fuzzy MCDM/MCDA methods are particularly important as they allow capturing the uncertainty and imprecision of the information used in making decisions. This method is the Fuzzy Preference Ranking Organization Method for Enrichment Evaluation (Fuzzy PROMETHEE), and its extension in the form of New Easy Approach to Fuzzy PROMETHEE (NEAT F-PROMETHEE). However, the unavailability of software using the NEAT F-PROMETHEE method significantly reduces its ease of use and may discourage potential users and researchers considering using it in their studies. Therefore, to facilitate the use of this MCDA method, the article presents the implementation of NEAT F-PROMETHEE in the MATLAB environment. Moreover, the verification of the developed implementation and its application in the management decision-making problem is presented, together with the analysis of the operation of the mapping correction function used in NEAT F-PROMETHEE. The results obtained with NEAT F-PROMETHEE were compared with the results of the Fuzzy PROMETHEE method which did not apply correction. The analysis shows that the correction applied in NEAT F-PROMETHEE allows obtaining results with a smaller error than the non-corrected implementations of PROMETHEE Fuzzy. Therefore, a more accurate solution of the decision problem is obtained.

- improving the process of mapping fuzzy numbers in the Fuzzy PROMETHEE method
- implementing a correction mechanism while mapping trapezoidal fuzzy numbers

A R T I C L E I N F O

Method name: PROMETHEE – Preference Ranking Organization METHod for Enrichment Evaluation
Keywords: Multi-criteria decision making, MATLAB, NEAT F-PROMETHEE, Uncertainty, Fuzzy sets, Outranking relation
Article history: Received 13 February 2021; Accepted 5 April 2021; Available online 13 April 2021

DOI of original article: 10.1016/j.eswa.2021.114686
E-mail address: pawel.ziemba@usz.edu.pl

https://doi.org/10.1016/j.mex.2021.101344

Specifications table

Subject Area	Computer Science
More specific subject area	*Decision Support System in Management*
Method name	*PROMETHEE – Preference Ranking Organization METHod for Enrichment Evaluation*
Name and reference of original method	*Brans, J.P., & De Smet, Y. (2016). PROMETHEE Methods. In S. Greco, M. Ehrgott, & J.R. Figueira (Eds.), Multiple Criteria Decision Analysis. State of the Art Surveys (2nd ed.) (pp. 187–219). New York, Springer* [2] *Geldermann, J., Spengler, T., & Rentz, O. (2000). Fuzzy outranking for environmental assessment. Case study: iron and steel making industry. Fuzzy Sets and Systems, 115(1), 45–65.* doi:10.1016/S0165-0114(99)00021-4 [3]

Method details

Decision-making is inseparable from management [4], and some researchers even claim that it is the most important function of management [5] and the most important task of managers [6]. In turn, most of the nontrivial management decision-making problems are of a multi-criteria nature, and Multi-criteria Decision Making/Aid/Analysis (MCDM/MCDA) methods in the fuzzy or crisp forms are used to solve them. Fuzzy methods, unlike crisp methods, allow to capturing uncertainty and imprecision [7], usually occurring in management decisions [6]. Therefore, fuzzy methods are widely used in management [8]. One of the MCDA methods often used in management problems [9] is the Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) [2], and its fuzzy or stochastic developments [10,11]. This is due to the ease of use and universality of the PROMETHEE method, transparency of its calculation procedure and high usefulness of the fuzzy versions of PROMETHEE in decision-making problems characterized by uncertainty [10,12,13]. The method which is a fuzzy development of PROMETHEE is New Easy Approach to Fuzzy PROMETHEE (NEAT F-PROMETHEE). This method, based on trapezoidal fuzzy numbers, has similar transparency and ease of use as the sharp PROMETHEE method. Moreover, it meets the methodological assumptions of the original PROMETHEE method, and thus gives the possibility to apply preference functions according to crisp PROMETHEE, as well as retaining appropriate scales for preference degrees and outranking flows. NEAT F-PROMETHEE uses six different preference functions, allows the use of linguistic, crisp and fuzzy values, in their natural scales, and gives the possibility of obtaining partial and total order of alternatives, thus offering great versatility. Finally, by applying the correction in the preference functions, it reduces the approximation errors that arise in other fuzzy PROMETHEE implementations when mapping fuzzy deviation to the form of a unicriterion preference degree. As a result, the NEAT F-PROMETHEE method is widely used in solving management decision problems [1,10,14,15]. On the other hand, the unavailability of software using the NEAT F-PROMETHEE method significantly reduces its ease of use and discourages potential users and researchers considering using NEAT F-PROMETHEE in their studies. Therefore, an important practical issue is the implementation of the NEAT F-PROMETHEE method in the programming language commonly used by researchers.

This article presents theoretical basis, technical details and MATLAB implementation of the NEAT F-PROMETHEE method. In the following part of the article there is a description of calculation procedures used in the method together with relevant mathematical equations and codes implementing these procedures in the MATLAB environment. The article ends with both the validation of the method based on the application of the developed MATLAB implementation in order to solve the decision problem and the analysis of obtained results.

Input data

The NEAT F-PROMETHEE method is a discrete MCDA method that addresses the problem of ranking m of fuzzy decision alternatives belonging to the set $\tilde{A} = \{\widetilde{a_1}, \widetilde{a_2}, \ldots, \widetilde{a_m}\}$ using n criteria, belonging to the set $C = \{c_1, c_2, \ldots, c_n\}$. It is based on trapezoidal fuzzy numbers (TFNs) in the form of $\widetilde{FN} =$

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		C1				C2				C3				C4				C5				C6			
2		FN1	FN2	FN3	FN4	FN1	FN2	FN3	FN4	FN1	FN2	FN3	FN4	FN1	FN2	FN3	FN4	FN1	FN2	FN3	FN4	FN1	FN2	FN3	FN4
3	A1	11.4	11.4	11.4	11.4	135	135	135	135	95	100	105	109	225	225	225	225	41	41	41	41	5	5	5	5
4	A2	13.2	13.2	13.2	13.2	135	135	135	135	92	92	92	92	220	220	220	220	41	41	41	41	5	5	5	5
5	A3	12.7	12.7	12.7	12.7	130	130	130	130	82	82	82	82	160	160	160	160	16.7	16.7	16.7	16.7	4	4	4	4

Fig. 1. The structure of the alternatives.xlsx file containing the values of alternatives for individual criteria.

(FN_1, FN_2, FN_3, FN_4), for which the membership function is described by the formula (1):

$$\mu_{\widetilde{FN}}(x) = \begin{cases} \dfrac{x - FN_1}{FN_2 - FN_1} \Leftrightarrow FN_1 \le x < FN_2 \\ 1 \Leftrightarrow FN_2 \le x \le FN_3 \\ \dfrac{x - FN_4}{a_3 - FN_4} \Leftrightarrow FN_3 < x \le FN_4 \\ 0 \Leftrightarrow otherwise \end{cases} \tag{1}$$

Firstly, the fuzzy weights of the criteria are obtained $\widetilde{Wf} = \{\widetilde{wf}_1, \widetilde{wf}_2, \ldots, \widetilde{wf}_n\}$ and the fuzzy values of the alternatives for each criterion. Both weights and values of alternatives can be expressed on natural scales of specific criteria (e.g. PLN or $ for Price), or on linguistic scales. The following is the content of the *LingVal.m* file, which defines the linguistic scales for weights of criteria and values of alternatives.

LingVal.m

```
%Linguistic variables for weights of criteria:
VL=[0 0 0.1 0.2];%Very low
L=[0.1 0.2 0.2 0.3];%Low
ML=[0.2 0.3 0.4 0.5];%Medium low
M=[0.4 0.5 0.5 0.6];%Medium
MH=[0.5 0.6 0.7 0.8];%Medium high
H=[0.7 0.8 0.8 0.9];%High
VH=[0.8 0.9 1.0 1.0];%Very High

%Linguistic variables for performances of alternatives:
VP=[0,0,1,2];% Very Poor
P=[1,2,2,3];% Poor
MP=[2,3,4,5];% Medium Poor
F=[4,5,5,6];% Fair
MG=[5,6,7,8];% Medium Good
G=[7,8,8,9];% Good
VG=[8,9,10,10];% Very Good
```

In turn, Fig. 1 shows the structure of the file *alternatives.xlsx*, from which the values of alternatives are loaded into the performance matrix.

The structure of the performance matrix E is described by the formula (2):

$$E = \begin{matrix} \\ \widetilde{a}_1 \\ \widetilde{a}_2 \\ \vdots \\ \widetilde{a}_m \end{matrix} \begin{bmatrix} c_1 & c_2 & \cdots & c_n \\ \widetilde{e}_{1,1} & \widetilde{e}_{1,2} & \cdots & \widetilde{e}_{1,n} \\ \widetilde{e}_{2,1} & \widetilde{e}_{2,2} & \cdots & \widetilde{e}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{e}_{m,1} & \widetilde{e}_{m,2} & \cdots & \widetilde{e}_{m,n} \end{bmatrix} \tag{2}$$

where $\widetilde{e_{i,j}} = c_j(\widetilde{a}_i)$, $\widetilde{e_{i,j}} = (e_{i,j_1}, e_{i,j_2}, e_{i,j_3}, e_{i,j_4}) = (c_j(a_{i1}), c_j(a_{i2}), c_j(a_{i3}), c_j(a_{i4}))$, therefore $\widetilde{e_{i,j}}$ represents the performance level of an alternative $\widetilde{a}_i$ according to a criterion c_j.

In addition to the values of the alternatives and the weights of the criteria, preference directions are defined at the beginning (for the 'profit' criteria, the maximum is preferred and for the 'cost'

criteria the minimum is preferred), as well as preference functions and thresholds related to the preference functions (indifference (q), preference (p), Gaussian (s)) for individual criteria are also defined. As a result, a complete model of the decision maker's preferences and, more broadly, a model of the decision-making problem is constructed. For the model developed in this way, in subsequent stages, calculations of the NEAT F-PROMETHEE method are performed, alternatives are ranked and results are displayed. The script code, including the indicated actions, is presented below.

main_script.m

```
%Variables used
%%NoOfCriteria - number of criteria
%%NoOfAlternatives - number of alternatives
%%E - decision matrix cell
%%Wf - fuzzy weights vector cell
%%PrefDirection - preference directions vector
%%PrefFun - preference functions vector
%%q - indifference thresholds vector
%%p - preference thresholds vector
%%s - Gaussian thresholds vector
clear all;
close all;
%Decision problem model
%%Decision-maker preference model
lingVal;%Linguistic values definitions
NoOfAlternatives=3;%Number of alternatives
NoOfCriteria=6;%Number of criteria
Wf={H H H H H H};%Weights of criteria
PrefDirection=[2 1 1 1 1 1];%Preference direction (1-max;2-min)
PrefFun=[5 5 5 5 3 1];%Preference functions
q=[0 0 0 0 0 0];%Indifference thresholds
p=[0 0 0 0 0 0];%Preference thresholds
s=[0 0 0 0 0 0];%Gaussian thresholds
%%Alternative performance model
[data,names,~]=xlsread('alternatives.xlsx','values','A3:Y5');%read data from file
names=names(:,1);%define alternative names
E=cell(NoOfAlternatives,NoOfCriteria);%performance matrix initiation
for i=1:NoOfAlternatives
    j=1;
    for k=1:4:NoOfCriteria*4
        E{i,j}=data(i,k:k+3);%performance matrix completing
        j=j+1;
    end
end
%NEAT F-PROMETHEE computations
[crispPhi,crispPhiPlus,crispPhiMinus,Phi,PhiPlus,PhiMinus,Pi,W,P,d]=NEAT_FPROMETHEE(NoOfCriteria
,NoOfAlternatives,E,Wf,PrefDirection,PrefFun,q,p,s);
%Rank alternatives
rankPhi=genRanking(round(crispPhi,8));
rankPhiPlus=genRanking(round(crispPhiPlus,8));
rankPhiMinus=genRanking(1-round(crispPhiMinus,8));
%Print results
showResults(NoOfAlternatives,names,rankPhi,rankPhiPlus,rankPhiMinus,crispPhi,crispPhiPlus,crispP
hiMinus);
plotResults(NoOfAlternatives,names,rankPhi,rankPhiPlus,rankPhiMinus,crispPhi,crispPhiPlus,crispP
hiMinus,Phi,PhiPlus,PhiMinus);
plotPartialOrder(NoOfAlternatives,names,rankPhiPlus,rankPhiMinus);
```

Calculations of NEAT F-PROMETHEE

Calculations for the NEAT F-PROMETHEE method are performed in several steps. First, the performance matrix E is transformed in such a way that the direction of preferences for each criterion is maximum. For this purpose, the values of the 'cost' criteria are transformed according to the formulae (3)(4):

$$\widetilde{e_{i,j}} = \widetilde{e_{i,j}} \times (-1) = \left(-e_{i,j_1}, -e_{i,j_2}, -e_{i,j_3}, -e_{i,j_4}\right) \tag{3}$$

$$\widetilde{e_{i,j}} = \left(-e_{i,j_4}, -e_{i,j_3}, -e_{i,j_2}, -e_{i,j_1}\right) \tag{4}$$

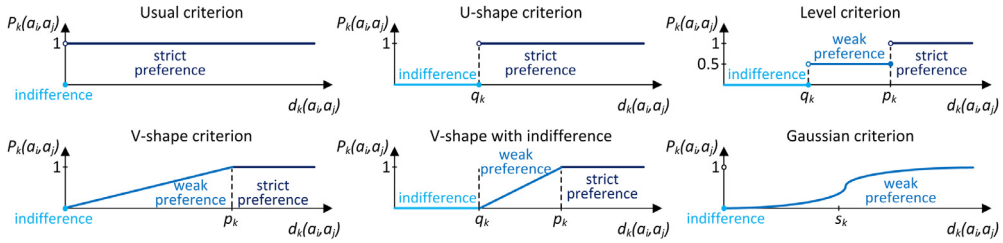

Fig. 2. Preference functions of the PROMETHEE method.

In the subsequent steps, fuzzy deviations are calculated and mapped to the form of unicriterion preference degrees and the defuzzification and normalisation of weights, preference aggregation and calculation of fuzzy outranking flows and defuzzification of calculated outranking flows take place. The code of the main NEAT F-PROMETHEE function is presented below.

NEAT_FPROMETHEE.m

```
function
[crispPhi,crispPhiPlus,crispPhiMinus,Phi,PhiPlus,PhiMinus,Pi,W,P,d]=NEAT_FPROMETHEE(NoOfCriteria
,NoOfAlternatives,E,Wf,PrefDirection,PrefFun,q,p,s)
%NEAT F-PROMETHEE function
    %if preference direction is min then multiply FN by -1 and flip FN
    for i=1:NoOfAlternatives
        for j=1:NoOfCriteria
            if PrefDirection(j)==2
                E{i,j}(1:4)=E{i,j}(1:4).*-1;
                E{i,j}(1:4)=fliplr(E{i,j});
            end
        end
    end
    %computation of fuzzy devation and mapping to unicriterion preference degrees
    [P,d]=mapDeviation(NoOfAlternatives,NoOfCriteria,E,PrefFun,q,p,s);
    %defuzzification of weights and normalisation to 1
    W=defuzWeight(NoOfCriteria,Wf);
    %preference aggregation
    [Phi,PhiPlus,PhiMinus,Pi]=aggrPreference(NoOfAlternatives,NoOfCriteria,P,W);
    %defuzzification of outranking flows
    [crispPhi,crispPhiPlus,crispPhiMinus]=defuzPhi(NoOfAlternatives,Phi,PhiPlus,PhiMinus);
end
```

The calculation of fuzzy deviations is carried out for each pair of alternatives for each criterion. It shall be carried out according to the formula (5):

$$\Lambda_{\widetilde{a}_i,\widetilde{a}_j\in\widetilde{A}}\Lambda_{c_k\in C}\widetilde{d_k}\left(\widetilde{a}_i,\widetilde{a}_j\right)=c_k(\widetilde{a}_i)\ominus c_k\left(\widetilde{a}_j\right)=\widetilde{e_{i,k}}\ominus\widetilde{e_{j,k}}=\left(e_{i,k_1},e_{i,k_2},e_{i,k_3},e_{i,k_4}\right)\ominus\left(e_{j,k_1},e_{j,k_2},e_{j,k_3},e_{j,k_4}\right)$$
$$=\left(e_{i,k_1}-e_{j,k_4},e_{i,k_2}-e_{j,k_3},e_{i,k_3}-e_{j,k_2},e_{i,k_4}-e_{j,k_1}\right) \quad (5)$$

Then, the fuzzy deviations obtained are mapped using the appropriate preference function. In the classic crisp PROMETHEE method, six preference functions are used, shown in Fig. 2.

The mapping of crisp numbers consists in calculating the value of the preference function P_k for the deviation d_k, so the function $P_k(d_k)$ is calculated. For TFNs, four values of the deviation $\widetilde{d_k}=(d_{k1},d_{k2},d_{k3},d_{k4})$ are already mapped and thus four values are obtained $\widetilde{P_k}(\widetilde{d_k})=(P_k(d_{k1}),P_k(d_{k2}),P_k(d_{k3}),P_k(d_{k4}))$. The comparison of the mapping of crisp numbers and TFNs is shown in Fig. 3.

In the case of crisp numbers, used in the classic PROMETHEE method, the preference functions allow precise mapping of the deviation value d_k to the form of the unicriterion preference degree $P_k(d_k)$. But in the case of TFNs, used in many fuzzy versions of PROMETHEE, approximation errors can occur during mapping. Therefore, the NEAT F-PROMETHEE method extends the mapping process with a function to correct mapping errors. The preference functions together with the correction functions used in NEAT F-PROMETHEE are shown in the formulae 6–(17).

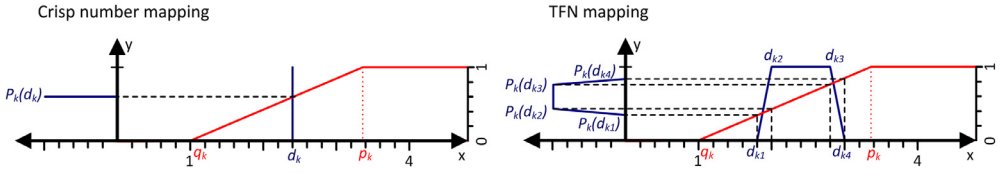

Fig. 3. The mapping of a crisp number and a TFN.

Usual criterion (6):

$$\widetilde{P}_k\left(\widetilde{d}_k\right) = \left(P_k(d_{k1}), P_k(d_{k2}), P_k(d_{k3}), P_k(d_{k4})\right) = \begin{cases} 0 \Leftrightarrow d_{k_l} \leq 0 \\ 1 \Leftrightarrow d_{k_l} > 0 \end{cases}, \; l = 1, \ldots, 4 \tag{6}$$

Correction for usual criterion (7):

$$\begin{cases} P_k(d_{k2}) = 0 \Leftarrow \dfrac{-d_{k1}}{d_{k2} - d_{k1}} > 0.5 \wedge d_{k1} < 0 \leq d_{k2} \\ P_k(d_{k3}) = 1 \Leftarrow \dfrac{-d_{k4}}{d_{k3} - d_{k4}} > 0.5 \wedge d_{k3} \leq 0 < d_{k4} \end{cases} \tag{7}$$

U-shaped criterion (8):

$$\widetilde{P}_k\left(\widetilde{d}_k\right) = \left(P_k(d_{k1}), P_k(d_{k2}), P_k(d_{k3}), P_k(d_{k4})\right) = \begin{cases} 0 \Leftrightarrow d_{k_l} \leq q_k \\ 1 \Leftrightarrow d_{k_l} > q_k \end{cases}, \; l = 1, \ldots, 4 \tag{8}$$

Correction for U-shaped criterion (9):

$$\begin{cases} P_k(d_{k2}) = 0 \Leftarrow \dfrac{q_k - d_{k1}}{d_{k2} - d_{k1}} > 0.5 \wedge d_{k1} < q_k \leq d_{k2} \\ P_k(d_{k3}) = 1 \Leftarrow \dfrac{q_k - d_{k4}}{d_{k3} - d_{k4}} > 0.5 \wedge d_{k3} \leq q_k < d_{k4} \end{cases} \tag{9}$$

V-shaped criterion (10):

$$\widetilde{P}_k\left(\widetilde{d}_k\right) = \left(P_k(d_{k1}), P_k(d_{k2}), P_k(d_{k3}), P_k(d_{k4})\right) = \begin{cases} 0 \Leftrightarrow d_{k_l} \leq 0 \\ \dfrac{d_{k_l}}{p_k} \Leftrightarrow 0 < d_{k_l} \leq p_k \\ 1 \Leftrightarrow d_{k_l} > p_k \end{cases}, \; l = 1, \ldots, 4 \tag{10}$$

Correction for V-shaped criterion (11):

$$\begin{cases} P_k(d_{k2}) = 0 \Leftarrow \dfrac{-d_{k1}}{d_{k2} - d_{k1}} > 0.5 \wedge d_{k1} < 0 \leq d_{k2} \\ P_k(d_{k3}) = 1 \Leftarrow \dfrac{p_k - d_{k4}}{d_{k3} - d_{k4}} > 0.5 \wedge d_{k3} \leq p_k < d_{k4} \end{cases} \tag{11}$$

Level criterion (12):

$$\widetilde{P}_k\left(\widetilde{d}_k\right) = \left(P_k(d_{k1}), P_k(d_{k2}), P_k(d_{k3}), P_k(d_{k4})\right) = \begin{cases} 0 \Leftrightarrow d_{k_l} \leq q_k \\ \dfrac{1}{2} \Leftrightarrow q_k < d_{k_l} \leq p_k \\ 1 \Leftrightarrow d_{k_l} > p_k \end{cases}, \; l = 1, \ldots, 4 \tag{12}$$

Correction for level criterion (13):

$$\begin{cases} P_k(d_{k2}) = 0 \Leftarrow \dfrac{q_k - d_{k1}}{d_{k2} - d_{k1}} > 0.5 \wedge d_{k1} < q_k \leq d_{k2} \\ P_k(d_{k3}) = 1 \Leftarrow \dfrac{p_k - d_{k4}}{d_{k3} - d_{k4}} > 0.5 \wedge d_{k3} \leq p_k < d_{k4} \end{cases} \tag{13}$$

V-shaped criterion with indifference area (14):

$$\widetilde{P}_k\left(\widetilde{d}_k\right) = \left(P_k(d_{k1}), P_k(d_{k2}), P_k(d_{k3}), P_k(d_{k4})\right) = \begin{cases} 0 \Leftrightarrow d_{k_l} \leq q_k \\ \dfrac{d_{k_l} - q_k}{p_k - q_k} \Leftrightarrow q_k < d_{k_l} \leq p_k, \ l = 1, \ldots, 4 \\ 1 \Leftrightarrow d_{k_l} > p_k \end{cases} \tag{14}$$

Correction for V-shaped criterion with indifference area (15):

$$\begin{cases} P_k(d_{k2}) = 0 \Leftarrow \dfrac{q_k - d_{k1}}{d_{k2} - d_{k1}} > 0.5 \wedge d_{k1} < q_k \leq d_{k2} \\ P_k(d_{k3}) = 1 \Leftarrow \dfrac{p_k - d_{k4}}{d_{k3} - d_{k4}} > 0.5 \wedge d_{k3} \leq p_k < d_{k4} \end{cases} \tag{15}$$

Gaussian criterion (16):

$$\widetilde{P}_k\left(\widetilde{d}_k\right) = \left(P_k(d_{k1}), P_k(d_{k2}), P_k(d_{k3}), P_k(d_{k4})\right) = \begin{cases} 0 \Leftrightarrow d_{k_l} \leq 0 \\ 1 - \exp\left(\dfrac{-{d_{k_l}}^2}{2{s_k}^2}\right) \Leftrightarrow d_{k_l} > 0 \end{cases}, \ l = 1, \ldots, 4 \tag{16}$$

Correction for Gaussian criterion (17):

$$P_k(d_{k2}) = 0 \Leftarrow \frac{-d_{k1}}{d_{k2} - d_{k1}} > 0.5 \wedge d_{k1} < 0 \leq d_{k2} \tag{17}$$

Fig. 4 shows an example of an approximation error that occurs during the TFN mapping, the correct mapping result and the operation of the correction mechanism used in the NEAT F-PROMETHEE method.

Fig. 4 contains a TFN $\widetilde{d}_k = (d_{k1}, d_{k2}, d_{k3}, d_{k4}) = (1, 2, 3, 4)$ mapped with the use of the linear preference function, called V-shaped criterion with indifference area, with parameters $q_k = 1.8$, $p_k = 3.4$. For TFN mapping, according to the formula (14), the mapping result will be a TFN $\widetilde{P}_k(\widetilde{d}_k) = (P_k(d_{k1}), P_k(d_{k2}), P_k(d_{k3}), P_k(d_{k4})) = (0, \frac{2-1.8}{3.4-1.8}, \frac{3-1.8}{3.4-1.8}, 1) = (0, 0.125, 0.75, 1)$. According to the membership function defined for a TFN [16,17], the fuzzy number values at the indicated points are $\mu_{\widetilde{P}_k(\widetilde{d}_k)}(y) = (0, 1, 1, 0)$. In the case of precision mapping, since none of the preference functions used is an injection (so the preference function can take the same values for two different values on the x-axis), the mapping function described by the formula (18) [18,19] should be used to determine the value of the fuzzy number in points (0,0.125,0.75,1).

$$\mu_{\widetilde{P}_k(\widetilde{d}_k)}(y) = \max_{y = P_k(x)} \mu_{\widetilde{d}_k}(x) \tag{18}$$

Based on the formula (18) and the maximum values $\widetilde{d}_k$ in points q_k and p_k, which, based on the formula (1), are $\mu_{\widetilde{d}_k}(q_k) = (\frac{q_k - d_{k1}}{d_{k2} - d_{k1}} \Leftrightarrow d_{k1} < q_k \leq d_{k2}) = \frac{1.8-1}{2-1} = 0.8$ and $\mu_{\widetilde{d}_k}(p_k) = (\frac{d_{k4} - p_k}{d_{k4} - d_{k3}} \Leftrightarrow d_{k3} \leq p_k < d_{k4}) = \frac{4-3.4}{4-3} = 0.6$ it, is easy to see that the fuzzy number values in points $(0, 0.125, 0.75, 1)$ should be $\mu_{\widetilde{P}_k(\widetilde{d}_k)}(y) = (0.8, 1, 1, 0.6)$. Therefore, TNF mapping generates a relatively large approximation error. That is why, in the NEAT F-PROMETHEE method, a shape correction of the obtained TFN was introduced to make it as close as possible to the result of precise mapping. The code

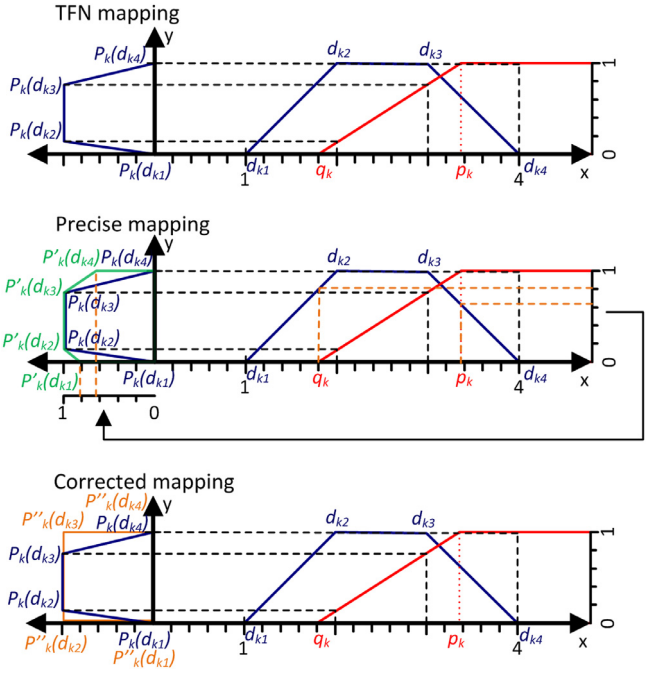

Fig. 4. TFN mapping, precise mapping and correction mapping.

of the MATLAB function, which calculates fuzzy deviation and the values of the correction preference function, is shown below.

mapDeviation.m

```
function [P,d]=mapDeviation(NoOfAlternatives,NoOfCriteria,E,PrefFun,q,p,s)
    %variables initiation
    d=cell(NoOfAlternatives,NoOfAlternatives,NoOfCriteria); d(:)={[0 0 0 0]};
    P=cell(NoOfAlternatives,NoOfAlternatives,NoOfCriteria); P(:)={[0 0 0 0]};
    %computation of fuzzy devation and mapping to unicriterion preference degrees
    for i=1:NoOfAlternatives
        for j=1:NoOfAlternatives
            if i~=j
                for k=1:NoOfCriteria
                    %fuzzy deviation
                    d{i,j,k}=E{i,k}-fliplr(E{j,k});
                    %usual criterion
                    if PrefFun(k)==1
                        for l=1:4
                            if d{i,j,k}(l)<=0
                                P{i,j,k}(l)=0;
                            else
                                P{i,j,k}(l)=1;
                            end
                        end
                    %correction
                    [d1,d2]=checkCorrection(d{i,j,k},0,0);
                    if d1
                            P{i,j,k}(2)=0;
                    end
                    if d2
                            P{i,j,k}(3)=1;
                    end
```

```
    %U-shape criterion
    elseif PrefFun(k)==2
        for l=1:4
            if d{i,j,k}(l)<=q(k)
                P{i,j,k}(l)=0;
            else
                P{i,j,k}(l)=1;
            end
        end
        %correction
        [d1,d2]=checkCorrection(d{i,j,k},q(k),q(k));
        if d1
                P{i,j,k}(2)=0;
        end
        if d2
                P{i,j,k}(3)=1;
        end
    %V-shape criterion
    elseif PrefFun(k)==3
        for l=1:4
            if d{i,j,k}(l)<=0
                P{i,j,k}(l)=0;
            elseif d{i,j,k}(l)>0 && d{i,j,k}(l)<=p(k)
                P{i,j,k}(l)=d{i,j,k}(l)/p(k);
            else
                P{i,j,k}(l)=1;
            end
        end
        %correction
        [d1,d2]=checkCorrection(d{i,j,k},0,p(k));
        if d1
                P{i,j,k}(2)=0;
        end
        if d2
                P{i,j,k}(3)=1;
        end
    %U-shape criterion
    elseif PrefFun(k)==4
        for l=1:4
            if d{i,j,k}(l)<=q(k)
                P{i,j,k}(l)=0;
            elseif d{i,j,k}(l)>q(k) && d{i,j,k}(l)<=p(k)
                P{i,j,k}(l)=0.5;
            else
                P{i,j,k}(l)=1;
            end
        end
        %correction
        [d1,d2]=checkCorrection(d{i,j,k},q(k),p(k));
        if d1
                P{i,j,k}(2)=0;
        end
        if d2
```

```
                        P{i,j,k}(3)=1;
                    end
                %V-shape with indifference area criterion
                elseif PrefFun(k)==5
                    for l=1:4
                        if d{i,j,k}(l)<=q(k)
                            P{i,j,k}(l)=0;
                        elseif d{i,j,k}(l)>q(k) && d{i,j,k}(l)<=p(k)
                            P{i,j,k}(l)=(d{i,j,k}(l)-q(k))/(p(k)-q(k));
                        else
                            P{i,j,k}(l)=1;

                        end
                    end
                    %correction
                    [d1,d2]=checkCorrection(d{i,j,k},q(k),p(k));
                    if d1
                            P{i,j,k}(2)=0;
                    end
                    if d2
                            P{i,j,k}(3)=1;
                    end
                %Gaussian criterion
                elseif PrefFun(k)==6
                    for l=1:4
                        if d{i,j,k}(l)<=0
                            P{i,j,k}(l)=0;
                        else
                            P{i,j,k}(l)=1-exp(-d{i,j,k}(l)^2/(2*s(k)^2));
                        end
                    end
                    %correction
                    [d1,~]=checkCorrection(d{i,j,k},0,Inf);
                    if d1
                            P{i,j,k}(2)=0;
                    end
                end
            end
        end
    end
end
```

A separate MATLAB function contains a procedure to check if a correction is required. From the formulae 6–(17), it can be seen that the conditions of correction for all preference functions are similar and can be recorded as the formula (19):

$$\begin{cases} P_k(d_{k2}) = 0 \Leftarrow \dfrac{t - d_{k1}}{d_{k2} - d_{k1}} > 0.5 \wedge d_{k1} < t \leq d_{k2} \\ P_k(d_{k3}) = 1 \Leftarrow \dfrac{u - d_{k4}}{d_{k3} - d_{k4}} > 0.5 \wedge d_{k3} \leq u < d_{k4} \end{cases} \tag{19}$$

However, the t and u variables allow distinguishing indifference, weak preference and strict preference relationships. Depending on the preference function used, t and u take different values:

- for the usual criterion: $t = 0$, $u = 0$,
- for the U-shaped criterion: $t = q_k$, $u = q_k$,
- for the V-shaped criterion: $t = 0$, $u = p_k$,
- for the level criterion and V-shaped criterion with indifference area: $t = q_k$, $u = p_k$,
- for the Gaussian criterion: $t = 0$, $u = \infty$.

In the case of $u = \infty$ for the Gaussian criterion, it should be clarified that this value is due to the property of the Gaussian preference function, which asymptotically tends to 1, and therefore does not allow obtaining strict preference [20]. In the case of $u = \infty$ there is an obvious contradiction because for a correction to occur, the value d_{k4} would have to be greater than infinity. Therefore, this correction function is not used for the Gaussian criterion. The code of the MATLAB function for checking the conditions of correction is as follows.

checkCorrection.m

```
function [out1,out2]=checkCorrection(d,t,u)
    if d(1)<t && d(2)>=t
        y=(t-d(1))/(d(2)-d(1));
        if y<=0.5
            out1=0;
        else
            out1=1;
        end
    else
        out1=0;
    end
    if d(3)<=u && d(4)>u
        y=(u-d(4))/(d(3)-d(4));
        if y<=0.5
            out2=0;
        else
            out2=1;
        end
    else
        out2=0;
    end
end
```

After the mapping and correction process, the weights of the criteria $\widetilde{wf}_j = (wf_1, wf_2, wf_3, wf_4)$ are defuzzified and normalised. As a result, a new vector of weights of criteria $W = \{w_1, w_2, \ldots, w_n\}$ is obtained. These actions are necessary to keep the scale $[-1,1]$ for the obtained solution, as it is done in the classic crisp PROMETHEE method. The defuzzification is performed using the Centroid method, described by the formula (20):

$$w_j = \frac{wf_{j_3}^2 + wf_{j_4}^2 + wf_{j_3} fw_{j_4} - wf_{j_1}^2 - wf_{j_2}^2 - wf_{j_1} wf_{j_2}}{3\left(wf_{j_3} + wf_{j_4} - wf_{j_1} - wf_{j_2}\right)} \tag{20}$$

The Centroid method, unlike the Bisector method, does not allow defuzzying crisp numbers (where $w_{j_1} = w_{j_2} = w_{j_3} = w_{j_4}$), so for such numbers a simple assignment $w_j = w_{j_1}$ should be used instead of the formula (20). The purpose of the normalisation is to bring the sum of all weights to 1 ($\sum_{j=1}^{n} w_j = 1$) and to define proportionally the weights of each criterion. It is performed according to the formula (21):

$$w_j = \frac{w_j}{\sum_{i=1}^{n} w_i} \tag{21}$$

The defuzzification and normalisation has been implemented in the appropriate MATLAB function.

defuzWeight.m

```
function W=defuzWeight(NoOfCriteria,Wf)
    %defuzzification of weights and normalisation to 1
    W=zeros(NoOfCriteria,1);
    for i=1:NoOfCriteria
        %Centroid method
        W(i)=(Wf{i}(3)^2+Wf{i}(4)^2+Wf{i}(3)*Wf{i}(4)-Wf{i}(1)^2-Wf{i}(2)^2-
Wf{i}(1)*Wf{i}(2))/(3*(Wf{i}(3)+Wf{i}(4)-Wf{i}(1)-Wf{i}(2)));
        %if weight is crisp then rewrite weight
        if isnan(W(i)) || isinf(W(i))
            W(i)=Wf{i}(1);
        end
    end
    %normalisation to 1
    tmp=sum(W);
    for i=1:NoOfCriteria
        W(i)=W(i)/tmp;
    end
end
```

In the next step, preferences are aggregated between the different pairs of decision alternatives and fuzzy outranking flows are calculated for each alternative. The aggregation of preferences is done according to the formula (22):

$$\Lambda_{\tilde{a}_i,\tilde{a}_j\in\tilde{A}\wedge w_k\in W}\tilde{\pi}\left(\tilde{a}_i,\tilde{a}_j\right)=\sum_{k=1}^{n}\tilde{P}_k\left(\tilde{d}_k\left(\tilde{a}_i,\tilde{a}_j\right)\right)\times w_k \tag{22}$$

After the aggregation of preferences, fuzzy outranking flows are calculated 23–(25):

$$\Lambda_{\tilde{a}_i\in\tilde{A}}\widetilde{\phi^+}(\tilde{a}_i)=\frac{\sum_{j=1,\,j\neq i}^{m}\tilde{\pi}\left(\tilde{a}_i,\tilde{a}_j\right)}{m-1} \tag{23}$$

$$\Lambda_{\tilde{a}_i\in\tilde{A}}\widetilde{\phi^-}(\tilde{a}_i)=\frac{\sum_{j=1,\,j\neq i}^{m}\tilde{\pi}\left(\tilde{a}_j,\tilde{a}_i\right)}{m-1} \tag{24}$$

$$\Lambda_{\tilde{a}_i\in\tilde{A}}\widetilde{\phi_{net}}(\tilde{a}_i)=\widetilde{\phi^+}(\tilde{a}_i)\ominus\widetilde{\phi^-}(\tilde{a}_i) \tag{25}$$

The given operations have been implemented as a function in MATLAB environment.

aggrPreference.m

```
function [Phi,PhiPlus,PhiMinus,Pi]=aggrPreference(NoOfAlternatives,NoOfCriteria,P,W)
    %variable initiation
    Pi=cell(NoOfAlternatives,NoOfAlternatives);
    Pi(:)={[0 0 0 0]};
    PhiPlus=cell(NoOfAlternatives,1);
    PhiPlus(:)={[0 0 0 0]};
    PhiMinus=cell(NoOfAlternatives,1);
    PhiMinus(:)={[0 0 0 0]};
    Phi=cell(NoOfAlternatives,1);
    Phi(:)={[0 0 0 0]};
    %global preference degrees
    for i=1:NoOfAlternatives
        for j=1:NoOfAlternatives
            if i~=j
                for k=1:NoOfCriteria
                    Pi{i,j}(1)=Pi{i,j}(1)+P{i,j,k}(1)*W(k);
```

```
                    Pi{i,j}(2)=Pi{i,j}(2)+P{i,j,k}(2)*W(k);
                    Pi{i,j}(3)=Pi{i,j}(3)+P{i,j,k}(3)*W(k);
                    Pi{i,j}(4)=Pi{i,j}(4)+P{i,j,k}(4)*W(k);
                end
            end
        end
    end
    %fuzzy positive and negative outranking flows
    for i=1:NoOfAlternatives
        for j=1:NoOfAlternatives
            PhiPlus{i}(1)=PhiPlus{i}(1)+Pi{i,j}(1)/(NoOfAlternatives-1);
            PhiPlus{i}(2)=PhiPlus{i}(2)+Pi{i,j}(2)/(NoOfAlternatives-1);
            PhiPlus{i}(3)=PhiPlus{i}(3)+Pi{i,j}(3)/(NoOfAlternatives-1);
            PhiPlus{i}(4)=PhiPlus{i}(4)+Pi{i,j}(4)/(NoOfAlternatives-1);
            PhiMinus{i}(1)=PhiMinus{i}(1)+Pi{j,i}(1)/(NoOfAlternatives-1);
            PhiMinus{i}(2)=PhiMinus{i}(2)+Pi{j,i}(2)/(NoOfAlternatives-1);
            PhiMinus{i}(3)=PhiMinus{i}(3)+Pi{j,i}(3)/(NoOfAlternatives-1);
            PhiMinus{i}(4)=PhiMinus{i}(4)+Pi{j,i}(4)/(NoOfAlternatives-1);
        end
    end
    %fuzzy net outranking flow
    for i=1:NoOfAlternatives
        Phi{i}(1)=PhiPlus{i}(1)-PhiMinus{i}(4);
        Phi{i}(2)=PhiPlus{i}(2)-PhiMinus{i}(3);
        Phi{i}(3)=PhiPlus{i}(3)-PhiMinus{i}(2);
        Phi{i}(4)=PhiPlus{i}(4)-PhiMinus{i}(1);
    end
end
```

The obtained values of fuzzy outranking flows are then defuzzified using the Centroid method 26–(28), similar to the fuzzy weights:

$$\Lambda_{\widetilde{a}_i\in\tilde{A}}\phi^+(\widetilde{a}_i)=\frac{\phi^+(\widetilde{a}_i)_3^2+\phi^+(\widetilde{a}_i)_4^2+\phi^+(\widetilde{a}_i)_3\phi^+(\widetilde{a}_i)_4-\phi^+(\widetilde{a}_i)_1^2-\phi^+(\widetilde{a}_i)_2^2-\phi^+(\widetilde{a}_i)_1\phi^+(\widetilde{a}_i)_2}{3(\phi^+(\widetilde{a}_i)_3+\phi^+(\widetilde{a}_i)_4-\phi^+(\widetilde{a}_i)_1-\phi^+(\widetilde{a}_i)_2)} \tag{26}$$

$$\Lambda_{\widetilde{a}_i\in\tilde{A}}\phi^-(\widetilde{a}_i)=\frac{\phi^-(\widetilde{a}_i)_3^2+\phi^-(\widetilde{a}_i)_4^2+\phi^-(\widetilde{a}_i)_3\phi^-(\widetilde{a}_i)_4-\phi^-(\widetilde{a}_i)_1^2-\phi^-(\widetilde{a}_i)_2^2-\phi^-(\widetilde{a}_i)_1\phi^-(\widetilde{a}_i)_2}{3(\phi^-(\widetilde{a}_i)_3+\phi^-(\widetilde{a}_i)_4-\phi^-(\widetilde{a}_i)_1-\phi^-(\widetilde{a}_i)_2)} \tag{27}$$

$$\Lambda_{\widetilde{a}_i\in\tilde{A}}\phi_{net}(\widetilde{a}_i)=\frac{\phi_{net}(\widetilde{a}_i)_3^2+\phi_{net}(\widetilde{a}_i)_4^2+\phi_{net}(\widetilde{a}_i)_3\phi_{net}(\widetilde{a}_i)_4-\phi_{net}(\widetilde{a}_i)_1^2-\phi_{net}(\widetilde{a}_i)_2^2-\phi_{net}(\widetilde{a}_i)_1\phi_{net}(\widetilde{a}_i)_2}{3(\phi_{net}(\widetilde{a}_i)_3+\phi_{net}(\widetilde{a}_i)_4-\phi_{net}(\widetilde{a}_i)_1-\phi_{net}(\widetilde{a}_i)_2)} \tag{28}$$

Similarly to the defuzzification of weights of criteria, if the outranking flows are crisp numbers (e.g. $\phi^+(\widetilde{a}_i)_1=\phi^+(\widetilde{a}_i)_2=\phi^+(\widetilde{a}_i)_3=\phi^+(\widetilde{a}_i)_4$), one should use a simple assignment (e.g. $\phi^+(\widetilde{a}_i)=\phi^+(\widetilde{a}_i)_1$). The MATLAB code responsible for defuzzification of outranking flows is shown below.

defuzPhi.m

```
function [crispPhi,crispPhiPlus,crispPhiMinus]=defuzPhi(NoOfAlternatives,Phi,PhiPlus,PhiMinus)
    %defuzzification of outranking flows
    for i=1:NoOfAlternatives
        %Phi Plus - centroid method
        crispPhiPlus(i,1)=(PhiPlus{i}(3)^2+PhiPlus{i}(4)^2+PhiPlus{i}(3)*PhiPlus{i}(4)-
PhiPlus{i}(1)^2-PhiPlus{i}(2)^2-PhiPlus{i}(1)*PhiPlus{i}(2))/(3*(PhiPlus{i}(3)+PhiPlus{i}(4)-
PhiPlus{i}(1)-PhiPlus{i}(2)));
        %Phi Minus - centroid method
        crispPhiMinus(i,1)=(PhiMinus{i}(3)^2+PhiMinus{i}(4)^2+PhiMinus{i}(3)*PhiMinus{i}(4)-
PhiMinus{i}(1)^2-PhiMinus{i}(2)^2-
PhiMinus{i}(1)*PhiMinus{i}(2))/(3*(PhiMinus{i}(3)+PhiMinus{i}(4)-PhiMinus{i}(1)-
PhiMinus{i}(2)));
        %Phi Net - centroid method
        crispPhi(i,1)=(Phi{i}(3)^2+Phi{i}(4)^2+Phi{i}(3)*Phi{i}(4)-Phi{i}(1)^2-Phi{i}(2)^2-
Phi{i}(1)*Phi{i}(2))/(3*(Phi{i}(3)+Phi{i}(4)-Phi{i}(1)-Phi{i}(2)));
        %if outranking flow is crisp then rewrite
        if isnan(crispPhiPlus(i,1)) || isinf(crispPhiPlus(i,1))
            crispPhiPlus(i,1)=PhiPlus{i}(1);
        end
        if isnan(crispPhiMinus(i,1)) || isinf(crispPhiMinus(i,1))
            crispPhiMinus(i,1)=PhiMinus{i}(1);
        end
        if isnan(crispPhi(i,1)) || isinf(crispPhi(i,1))
            crispPhi(i,1)=Phi{i}(1);
        end
    end
end
```

Generating rankings and displaying the results of the method

On the basis of the defuzzified values ϕ_{net}, a full NEAT F-PROMETHEE II (total order) ranking is generated, while the values ϕ^+ i ϕ^- are the basis for constructing the rankings that are later used in the NEAT F-PROMETHEE I (partial order) ranking. The MATLAB function responsible for this assigns each alternative an appropriate rank in the full ranking and the rankings ϕ^+ i ϕ^-.

genRanking.m

```
function ranking=genRanking(v)
    %if vector of alternative performances is not empty
    if ~isnan(v)
        %check vector length
        n=length(v);
        %ranking initiation
        ranking=zeros(1,n);
        %index initiation
        i=1;
        %while alternatives are not ranked
        while any(v~=-Inf)
            %find alternative with max performance
            tmp=find(v==max(v));
            %if one alternative was found, give it rank i and mark it in v
            if length(tmp)==1
                ranking(tmp)=i;
                v(tmp)=-Inf;
            %if multiple alternatives were found, give them rank of i and mark them in v
            else
                for j=1:length(tmp)
                    ranking(tmp(j))=i;
                    v(tmp(j))=-Inf;
                end
            end
            %increase index
            i=i+length(tmp);
        end
    else
        error('Vector v is empty');
    end
end
```

Table 1
Fuzzy assessments of alternatives.

Criterion	$\widetilde{a_1}$	$\widetilde{a_2}$	$\widetilde{a_3}$	$\widetilde{a_4}$
c_1 – Price [thousand PLN]	130	117	96	136
c_2 – Quality [Linguistic]	G	MG	G	F
c_3 – Green product [Linguistic]	G	MG	MP	MG
c_4 – Lead time [Days]	(28,30,30,32)	(24,26,30,30)	(20,22,24,26)	(22,22,24,24)
c_5 – Reliability [Linguistic]	G	G	MG	VG
c_6 – Green delivery [Linguistic]	P	G	VP	MP

Table 2
Preference model.

Criterion	Weight	Preference direction	Preference function	Indifference threshold (q)	Preference threshold (p)	Gaussian threshold (s)
c_1 – Price [thousand PLN]	H	Min	V-shape with indifference	5	40	–
c_2 – Quality [Linguistic]	H	Max	Usual	–	–	–
c_3 – Green product [Linguistic]	M	Max	Level	0	1	–
c_4 – Lead time [Days]	MH	Min	V-shape	–	5	–
c_5 – Reliability [Linguistic]	VH	Max	V-shape with indifference	0.5	1.5	–
c_6 – Green delivery [Linguistic]	ML	Max	Gaussian	–	–	1

After three rankings have been constructed, they are presented to the decision maker using the *showResults* function together with the defuzzifieded values of outranking flows (ϕ_{net}, ϕ^+, ϕ^-), which are the basis for building these rankings.

showResults.m

```
function
showResults(NoOfAlternatives,names,rankPhi,rankPhiPlus,rankPhiMinus,NcrispPhi,NcrispPhiPlus,Ncri
spPhiMinus)
    %results initiation
    resultsPhiNet=cell(NoOfAlternatives,3);
    resultsPhiPlus=cell(NoOfAlternatives,3);
    resultsPhiMinus=cell(NoOfAlternatives,3);
    %group all Phi Net results
    for i=1:NoOfAlternatives
        resultsPhiNet(i,:)=[names(i,:) NcrispPhi(i,:) rankPhi(i)];
    end
    %group all Phi Plus results
    for i=1:NoOfAlternatives
        resultsPhiPlus(i,:)=[names(i,:) NcrispPhiPlus(i,:) rankPhiPlus(i)];
    end
    %group all Phi Minus results
    for i=1:NoOfAlternatives
        resultsPhiMinus(i,:)=[names(i,:) NcrispPhiMinus(i,:) rankPhiMinus(i)];
    end
    %print results
    resultsPhiNet
    resultsPhiPlus
    resultsPhiMinus
end
```

In addition to presenting the rankings in the form of numerical values, in the NEAT F-PROMETHEE implementation, the results are also presented in a graphic form. This is performed by *plotResults* and

Table 3
The values of outranking flows and rankings of alternatives (NEAT F-PROMETHEE).

$\widetilde{A}$	ϕ^+	Rank ϕ^+	ϕ^-	Rank ϕ^-	ϕ_{net}	Rank NEAT F-PROMETHEE II (ϕ_{net})	Rank NEAT F-PROMETHEE I ($\phi^+ \circ \phi^-$)
$\widetilde{a}_1$	0.3368	4	0.3754	3	−0.0386	3	$\widetilde{a}_2 \succ \widetilde{a}_1$; $\widetilde{a}_3 \succ \widetilde{a}_1$
$\widetilde{a}_2$	0.3747	2	0.3301	1	0.0443	2	$\widetilde{a}_2 \succ \widetilde{a}_1$; $\widetilde{a}_2 \succ \widetilde{a}_4$
$\widetilde{a}_3$	0.4165	1	0.3721	2	0.0448	1	$\widetilde{a}_3 \succ \widetilde{a}_1$; $\widetilde{a}_3 \succ \widetilde{a}_4$
$\widetilde{a}_4$	0.3419	3	0.3906	4	−0.0489	4	$\widetilde{a}_2 \succ \widetilde{a}_4$; $\widetilde{a}_3 \succ \widetilde{a}_4$

plotPartialOrder functions. The *plotResults* function presents graphs ϕ_{net}, ϕ^+ i ϕ^- containing fuzzy and crisp outranking flows for individual alternatives, together with the positions of these alternatives in the rankings. The chart ϕ^+ shows how much a given alternative is outranking the others, while the chart ϕ^- depicts how much a given alternative is outranked by the others. In turn, the graph ϕ_{net} illustrates the total order of alternatives, in other words, it presents a solution to a decision-making problem using the NEAT F-PROMETHEE II method. It should be added that there may be two preference relationships in the total order of alternatives: (1) indifference between $\widetilde{a}_i$ and $\widetilde{a}_j$ ($\widetilde{a}_i \, I \, \widetilde{a}_j$) when $\phi_{net}(\widetilde{a}_i) = \phi_{net}(\widetilde{a}_j)$, (2) preference of $\widetilde{a}_i$ over $\widetilde{a}_j$ ($\widetilde{a}_i \succ \widetilde{a}_j$) when $\phi_{net}(\widetilde{a}_i) > \phi_{net}(\widetilde{a}_j)$.

plotResults.m

```
function
plotResults(NoOfAlternatives,names,rankPhi,rankPhiPlus,rankPhiMinus,NcrispPhi,NcrispPhiPlus,Ncri
spPhiMinus,NPhi,NPhiPlus,NPhiMinus)
    %generate perceptually-distinct colors
    colors=distinguishable_colors(NoOfAlternatives);
    %generate labels
    ylabels=cell(1,NoOfAlternatives);
    ticks=0:0.1:(NoOfAlternatives)/10;
    labels=NoOfAlternatives:-1:1;
    for i=1:NoOfAlternatives
        ylabels(i)=cellstr([num2str(labels(i),'%d')]);
    end
    ylabels=[{' '} ylabels {' '}];
    %Plot PhiNet (NEAT F-PROMETHEE II)
    %%generate figure with subplots
    figure;
    subplot(2,2,[3,4]);
    title('Phi_n_e_t fuzzy and crisp outranking flow');
    grid on;
    grid minor;
    hold on;
    yticks(ticks);
    yticklabels(ylabels);
    ylim([0 (NoOfAlternatives+0.5)*0.1]);
    %%show trapezoidal fuzzy numbers & crisp numbers
    for i=1:NoOfAlternatives
        leg(i)=plot(NPhi{i}(:),[0 (NoOfAlternatives+1-rankPhi(i))*0.1 (NoOfAlternatives+1-
rankPhi(i))*0.1 0],'Color',colors(i,:),'LineWidth',2);
        plot([NcrispPhi(i) NcrispPhi(i)],[0 (NoOfAlternatives+1-rankPhi(i))*0.1],'--
','Color',colors(i,:),'LineWidth',2);
    end
    %%show alternative names
    for i=1:NoOfAlternatives
        text(NcrispPhi(i),(NoOfAlternatives+1-
rankPhi(i))*0.1,names(i,:),'Color',colors(i,:),'BackgroundColor','none','FontWeight','bold','Hor
izontalAlignment','center','VerticalAlignment','bottom');
    end
    %%show legend and labels
    legend([leg(1:i)],names(1:i,:),'Location','eastoutside','Orientation','vertical');
    xlabel('\Phi_n_e_t');
    ylabel('Rank');
    %Plot PhiPlus
    subplot(2,2,1);
    title('Phi^+ fuzzy and crisp outranking flow');
    grid on;
    grid minor;
```

```
    hold on;
    yticks(ticks);
    yticklabels(ylabels);
    ylim([0 (NoOfAlternatives+0.5)*0.1]);
    for i=1:NoOfAlternatives
        leg(i)=plot(NPhiPlus{i}(:),[0 (NoOfAlternatives+1-rankPhiPlus(i))*0.1
(NoOfAlternatives+1-rankPhiPlus(i))*0.1 0],'Color',colors(i,:),'LineWidth',2);
        plot([NcrispPhiPlus(i) NcrispPhiPlus(i)],[0 (NoOfAlternatives+1-rankPhiPlus(i))*0.1],'--
','Color',colors(i,:),'LineWidth',2);
    end
    for i=1:NoOfAlternatives
        text(NcrispPhiPlus(i),(NoOfAlternatives+1-
rankPhiPlus(i))*0.1,names(i,:),'Color',colors(i,:),'BackgroundColor','none','FontWeight','bold',
'HorizontalAlignment','center','VerticalAlignment','bottom');
    end
    xlabel('\Phi^+');
    ylabel('Rank');
    %Plot PhiMinus
    subplot(2,2,2);
    title('Phi^- fuzzy and crisp outranking flow');
    grid on;
    grid minor;
    hold on;
    yticks(ticks);
    yticklabels(ylabels);
    ylim([0 (NoOfAlternatives+0.5)*0.1]);
    for i=1:NoOfAlternatives
        leg(i)=plot(NPhiMinus{i}(:),[0 (NoOfAlternatives+1-rankPhiMinus(i))*0.1
(NoOfAlternatives+1-rankPhiMinus(i))*0.1 0],'Color',colors(i,:),'LineWidth',2);
        plot([NcrispPhiMinus(i) NcrispPhiMinus(i)],[0 (NoOfAlternatives+1-
rankPhiMinus(i))*0.1],'--','Color',colors(i,:),'LineWidth',2);
    end
    for i=1:NoOfAlternatives
        text(NcrispPhiMinus(i),(NoOfAlternatives+1-
rankPhiMinus(i))*0.1,names(i,:),'Color',colors(i,:),'BackgroundColor','none','FontWeight','bold'
,'HorizontalAlignment','center','VerticalAlignment','bottom');
    end
    xlabel('\Phi^-');
    ylabel('Rank');
end
```

The *plotPartialOrder* function presents in a graphical form a partial order of alternatives, constructed on the basis of ϕ^+ and ϕ^- rankings. There may be three preference relationships in the partial order of alternatives: (1) indifference between $\widetilde{a}_i$ and $\widetilde{a}_j$ ($\widetilde{a}_i \, I \, \widetilde{a}_j$) when $(\widetilde{\phi^+}(\widetilde{a}_i) = \widetilde{\phi^+}(\widetilde{a}_j)) \wedge (\widetilde{\phi^-}(\widetilde{a}_i) = \widetilde{\phi^-}(\widetilde{a}_j))$, (2) preference of $\widetilde{a}_i$ over $\widetilde{a}_j$ ($\widetilde{a}_i \succ \widetilde{a}_j$) when $(\widetilde{\phi^+}(\widetilde{a}_i) \geq \widetilde{\phi^+}(\widetilde{a}_j)) \wedge (\widetilde{\phi^-}(\widetilde{a}_i) \leq \widetilde{\phi^-}(\widetilde{a}_j))$, where one of the inequalities is strict, (3) incomparability between $\widetilde{a}_i$ over $\widetilde{a}_j$ ($\widetilde{a}_i \, R \, \widetilde{a}_j$) when $(\widetilde{\phi^+}(\widetilde{a}_i) > \widetilde{\phi^+}(\widetilde{a}_j) \wedge \widetilde{\phi^-}(\widetilde{a}_i) > \widetilde{\phi^-}(\widetilde{a}_j)) \vee (\widetilde{\phi^+}(\widetilde{a}_i) < \widetilde{\phi^+}(\widetilde{a}_j) \wedge \widetilde{\phi^-}(\widetilde{a}_i) < \widetilde{\phi^-}(\widetilde{a}_j))$. The partial order presents an order of alternatives using the indicated preference relationships. It should be noted that the graphic presentation of the partial order shows indifference and preference

Table 4
The values of outranking flows and the rankings of alternatives obtained without mapping correction (Fuzzy PROMETHEE).

$\widetilde{A}$	ϕ^+	Rank ϕ^+	ϕ^-	Rank ϕ^-	ϕ_{net}	Rank Fuzzy PROMETHEE II (ϕ_{net})	Rank Fuzzy PROMETHEE I ($\phi^+ \circ \phi^-$)
$\widetilde{a}_1$	0.3320	4	0.37207	3	−0.0399	4	$\widetilde{a}_2 \succ \widetilde{a}_1$
$\widetilde{a}_2$	0.3784	2	0.3336	1	0.0441	1	$\widetilde{a}_2 \succ \widetilde{a}_1$
$\widetilde{a}_3$	0.4038	1	0.37209	2	0.0317	2	$\widetilde{a}_3 \succ \widetilde{a}_4$
$\widetilde{a}_4$	0.3522	3	0.3867	4	−0.0346	3	$\widetilde{a}_3 \succ \widetilde{a}_4$

Table 5
Mapping errors with and without correction.

Preference function	Error with correction	Error without correction	Improvement of the results [%]
Usual criterion	0.9956	1.3272	25%
V-shaped criterion	0.062	0.0879	29%
Level criterion	0.3864	0.607	36%
V-shaped criterion with indifference area	0.3388	0.6056	44%
Gaussian criterion	0.1467	0.1467	0%

relations in the form of edges connecting the alternatives directly or indirectly, while incomparability is represented by the lack of direct or indirect connection.

plotPartialOrder.m

```
function plotPartialOrder(NoOfAlternatives,names,rankPhiPlus,rankPhiMinus)
    %relation matrices initiation
    PreferenceRel=zeros(NoOfAlternatives);
    IndifferenceRel=zeros(NoOfAlternatives);
    UncomparabilityRel=zeros(NoOfAlternatives);
    %relation computations
    for i=1:NoOfAlternatives
        for j=1:NoOfAlternatives
            if i~=j
                if rankPhiPlus(i)<rankPhiPlus(j) && rankPhiMinus(i)<rankPhiMinus(j) || ...
                   rankPhiPlus(i)==rankPhiPlus(j) && rankPhiMinus(i)<rankPhiMinus(j) || ...
                   rankPhiPlus(i)<rankPhiPlus(j) && rankPhiMinus(i)==rankPhiMinus(j)
                    PreferenceRel(i,j)=1;
                elseif rankPhiPlus(i)==rankPhiPlus(j) && rankPhiMinus(i)==rankPhiMinus(j)
                    IndifferenceRel(i,j)=1;
                elseif rankPhiPlus(i)<rankPhiPlus(j) && rankPhiMinus(i)>rankPhiMinus(j) || ...
                       rankPhiPlus(i)>rankPhiPlus(j) && rankPhiMinus(i)<rankPhiMinus(j)
                    UncomparabilityRel(i,j)=1;
                end
            end
        end
    end
    Rank=sum(PreferenceRel)+1;
    %computate coordinates of alternatives on plot area
    coordinates=zeros(3,NoOfAlternatives);
    i=1;
    j=1;
    while min(Rank)<NoOfAlternatives+1
        alternative=find(Rank==min(Rank));
        for k=1:length(alternative)
            coordinates(:,i+j-1)=[NoOfAlternatives-j-k+1;i;alternative(k)];
            Rank(alternative(k))=NoOfAlternatives+1;
            if length(alternative)>1 && k<length(alternative)
                j=j+1;
            end
        end
        i=i+1;
    end
    coordinates2=coordinates;
    %Plot Partial Order (NEAT F-PROMETHEE I)
    %%generate figure
    figure;
    title('NEAT F-PROMETHEE I partial order');
    grid on;
    grid minor;
    hold on;
    axis([min(coordinates(2,:))-1 max(coordinates(2,:))+1 min(coordinates(1,:))-1
max(coordinates(1,:))+1]);
    xticklabels({''});
```

```
    yticklabels({''});
    %%show preference and indifference relations
    i=1;
    while min(coordinates(2,:))<NoOfAlternatives+1
        alternative=find(coordinates(2,:)==min(coordinates(2,:)));
        for j=1:length(alternative)
            l=0;
            for k=coordinates(2,alternative(j)):NoOfAlternatives
alternative2=find(coordinates(2,:)==min(setdiff(coordinates(2,:),min(coordinates(2,:)))));
                if PreferenceRel(coordinates(3,alternative(j)),coordinates(3,k)) &&
l<length(alternative2)
                    hP=plot([coordinates(2,alternative(j))+0.2 coordinates(2,k)-
0.2],[coordinates(1,alternative(j)) coordinates(1,k)],'-','Color','r','LineWidth',1);
                    line2arrow(hP);
                    l=l+1;
                end
                if IndifferenceRel(coordinates(3,alternative(j)),coordinates(3,k))==1 &&
coordinates(2,alternative(j))<=NoOfAlternatives && coordinates(2,k)<=NoOfAlternatives
                    hI=plot([coordinates(2,alternative(j))
coordinates(2,k)],[coordinates(1,alternative(j))-0.2 coordinates(1,k)+0.2],'-
.','Color','b','LineWidth',1);
                end
            end
            coordinates(2,alternative(j))=NoOfAlternatives+1;
        end
        i=i+1;
    end
    %%show alternative names and legend
    for i=1:NoOfAlternatives

text(coordinates2(2,i),coordinates2(1,i),names(coordinates2(3,i),:),'Color','b','BackgroundColor
','white','FontWeight','bold','HorizontalAlignment','center','VerticalAlignment','middle');
    end
    if sum(sum(PreferenceRel)) && sum(sum(IndifferenceRel))
        legend([hP
hI],{'Preference','Indifference'},'Location','southwest','Orientation','vertical');
    elseif sum(sum(PreferenceRel))
        legend(hP,'Preference','Location','southwest','Orientation','vertical');
    elseif sum(sum(IndifferenceRel))
        legend(hI,'Indifference','Location','southwest','Orientation','vertical');
    end
end
```

In the developed implementation of the NEAT F-PROMETHEE method, apart from the proprietary functions, the *distinguishable_colors* [21] and *line2arrow* [22] functions were also used.

Method validation

The correctness of the implementation of the NEAT F-PROMETHEE method has been verified by solving the decision problem on the basis of selecting a "green" supplier of electronic items for a manufacturing company in order to reduce costs at the manufacturing stage of finished products. In the decision-making process, 4 suppliers $\tilde{A} = \{\tilde{a_1}, \tilde{a_2}, \tilde{a_3}, \tilde{a_4}\}$, have been considered, assessing them against 6 criteria $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. Table 1 shows the parameters of the different decision alternatives and Table 2 includes the preference model used, i.e. the weights of the criteria, preference directions, preference functions and thresholds.

The application of the NEAT F-PROMETHEE method presented in the article has made it possible to obtain a solution to the decision problem, presented in Table 3 and Figs. 5 and 6.

Figs. 5 and 6, generated using *plotResults.m* and *plotPartialOrder.m* functions, enable an analysis of the obtained solution. Fig. 5 shows the fuzzy and disinfected values of alternatives, as well as the order of alternatives, separately for ϕ^+, ϕ^- and ϕ_{net} rankings. The ranking ϕ^+ allows us to conclude that the alternative, which is outranking all others the most, is $\tilde{a_3}$. It should be noted that the support of fuzzy numbers indicates that $\tilde{a_3}$ can be outranked in the ranking ϕ^+ by an alternative $\tilde{a_2}$, or even $\tilde{a_4}$. In turn, according to the ranking ϕ^-, the alternative most outranked by the others is $\tilde{a_4}$, although the analysis of fuzzy numbers indicates the possibility that the other alternatives will be outranked by $\tilde{a_4}$. Finally, when analysing the ranking ϕ_{net} the solution to the decision problem is

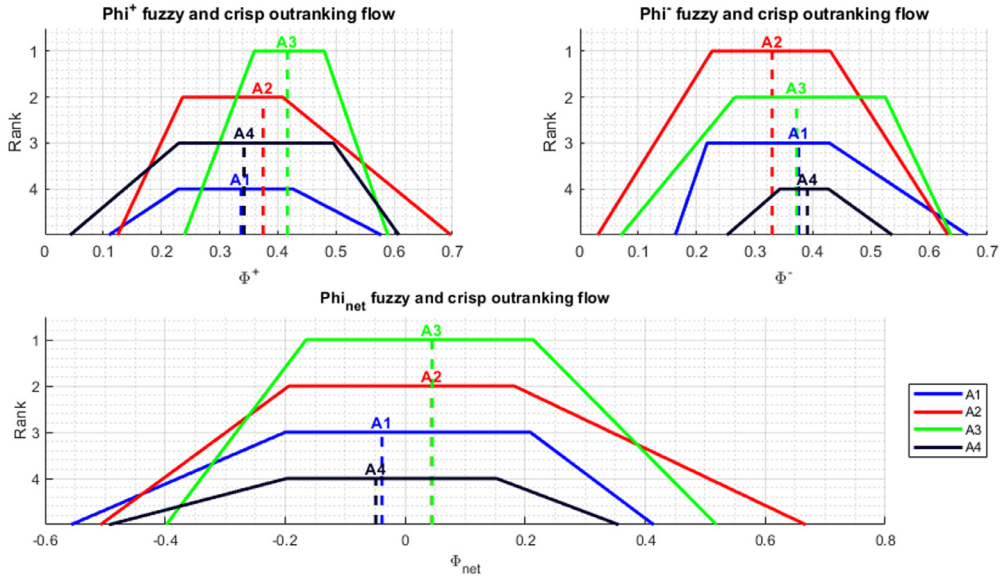

Fig. 5. Graphical representation of outranking flows (ϕ_{net}, ϕ^{+}, ϕ^{-}) and corresponding rankings (NEAT F-PROMETHEE II).

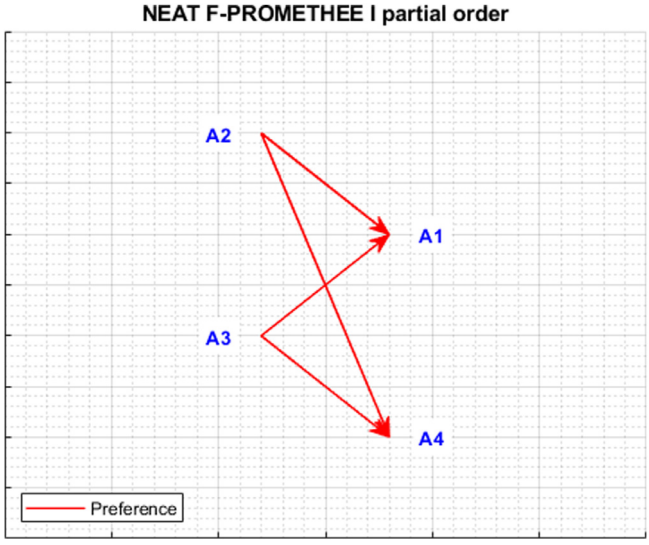

Fig. 6. Partial order NEAT F-PROMETHEE I.

the following total order of alternatives: $\widetilde{a_3} \succ \widetilde{a_2} \succ \widetilde{a_1} \succ \widetilde{a_4}$. However, the total order of alternatives obtained is characterised by a relatively high degree of uncertainty, as evidenced by the wide range of kernels and support for the fuzzy numbers obtained. As regards Fig. 6, which shows the partial order of the alternatives, it should be concluded that the dominant alternatives are $\widetilde{a_3}$ and $\widetilde{a_2}$, which are preferred over $\widetilde{a_1}$ and $\widetilde{a_4}$. This calculation example shows the usefulness of the fuzzy approach to interpret the degree of uncertainty of the solution obtained.

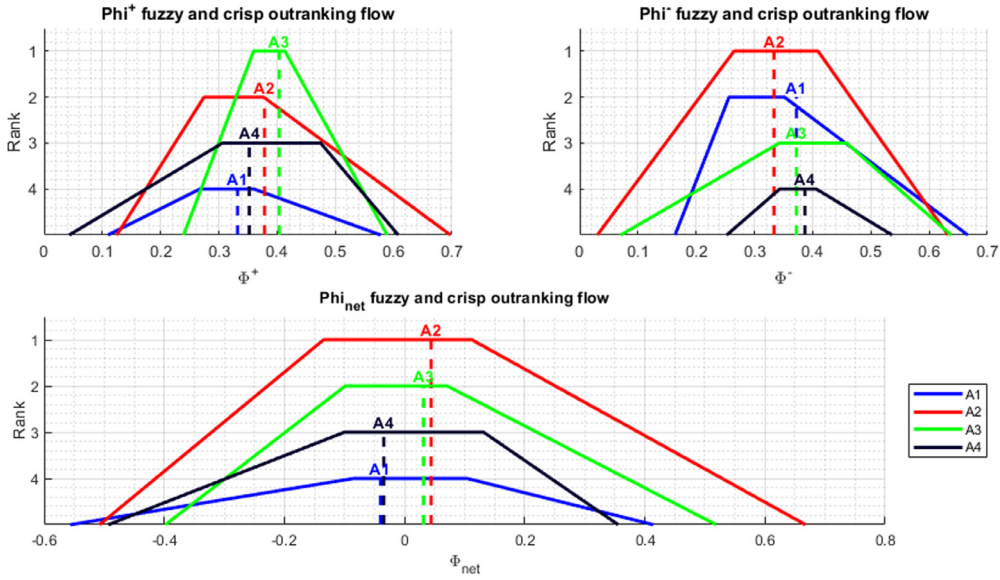

Fig. 7. Graphical representation of outranking flows (ϕ_{net}, ϕ^{+}, ϕ^{-}) and their corresponding rankings obtained without mapping correction (Fuzzy PROMETHEE II).

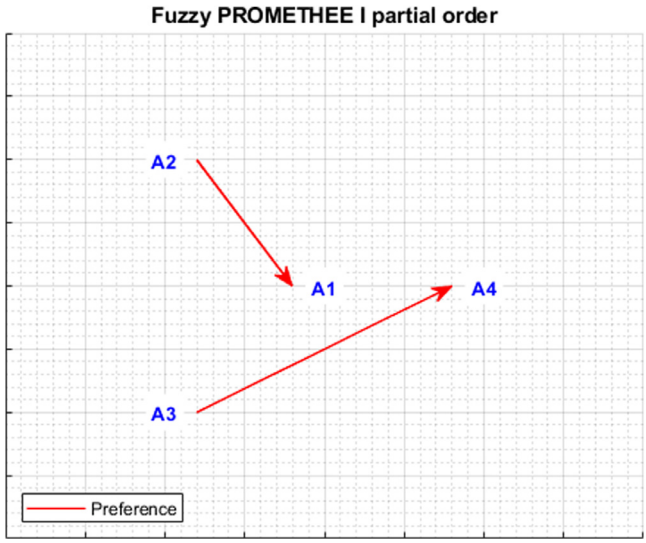

Fig. 8. Partial order obtained without mapping correction (Fuzzy PROMETHEE I).

Apart from the verification of the correctness of the implementation of the NEAT F-PROMETHEE method in the MATLAB environment, the operation of the correction of mapping errors (see Formulae (6)-(19)) and the impact of the correction on the obtained solution were also verified. For this purpose, the presented decision problem was solved using fuzzy PROMETHEE without correction. The solution obtained in this way is shown in Table 4 and Figs. 7 and 8.

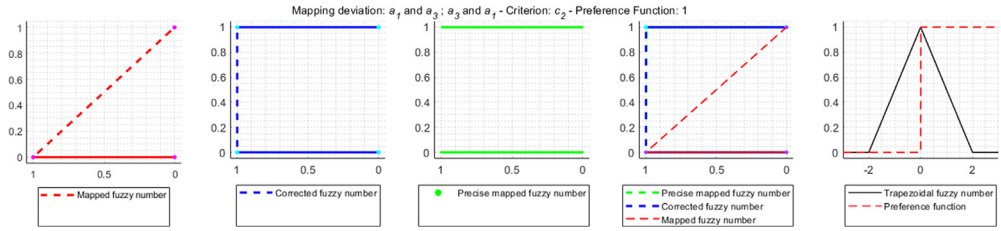

Fig. 9. Error and correction during the mapping of deviations $\widetilde{d}_2(\widetilde{a}_1, \widetilde{a}_3)$ and $\widetilde{d}_2(\widetilde{a}_3, \widetilde{a}_1)$.

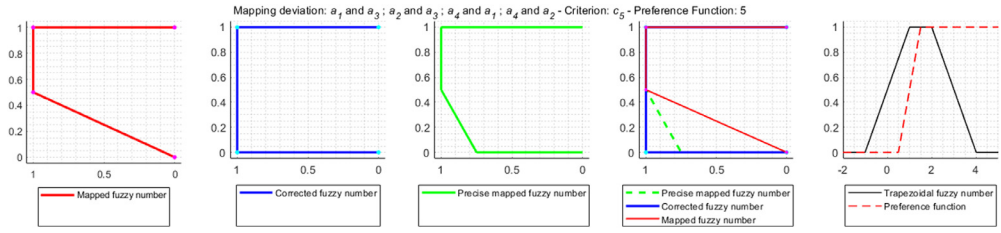

Fig. 10. Error and correction during the mapping of deviations $\widetilde{d}_5(\widetilde{a}_1, \widetilde{a}_3)$, $\widetilde{d}_5(\widetilde{a}_2, \widetilde{a}_3)$, $\widetilde{d}_5(\widetilde{a}_4, \widetilde{a}_1)$ and $\widetilde{d}_5(\widetilde{a}_4, \widetilde{a}_2)$.

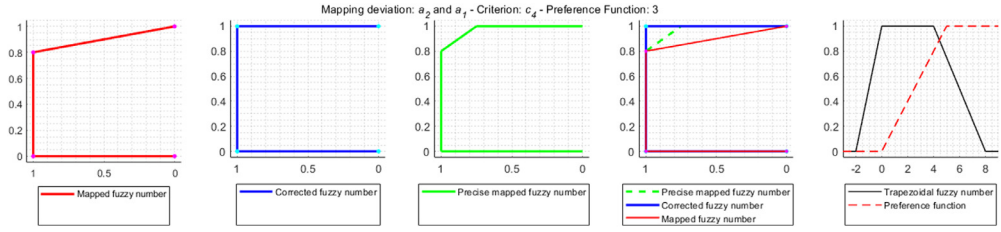

Fig. 11. Error and correction during the mapping of deviations $\widetilde{d}_4(\widetilde{a}_2, \widetilde{a}_1)$.

A comparison of Table 3 and Figs. 4 and 5 (solution with correction) with Table 4 and Figs. 6 and 7 (solution without correction) shows that, in the absence of correction, there was a change in the ranking ϕ_{net} in positions 1–2 and 3–4. In addition, the partial order of alternatives, namely preference relationships $\widetilde{a}_3 \succ \widetilde{a}_1$, $\widetilde{a}_2 \succ \widetilde{a}_4$ were converted into incomparability relationships $\widetilde{a}_1 \, R \, \widetilde{a}_3$, $\widetilde{a}_2 \, R \, \widetilde{a}_4$. In order to clearly determine which solution is correct, approximation errors resulting from the use of trapezoidal fuzzy numbers instead of accurate fuzzy mapping were examined. As a result of the study, it was found that in the decision problem under consideration, the approximation error occurs relatively often, because in 32 cases out of 72 mappings performed, i.e. in 44% of cases. On the other hand, the correction is made in 9 mappings, i.e. 12.5% of all cases and 28% of the mappings are affected by an error. The mappings for which the correction is made are shown in Figs. 9–12.

In addition to the mapping analysis during which correction is applied, cumulative mapping errors were also examined, with and without correction. Mapping errors were calculated for each of the preference functions, by defuzzyfing fuzzy numbers obtained using precise mapping (29) and trapezoidal fuzzy numbers (30) obtained using non-corrected and corrected mapping. Then the errors for each criterion were summed up, separately for each preference function P, where $P \in$ *{usual criterion, V-shaped criterion, level criterion, V-shaped criterion with indifference area, Gaussian criterion}* (31) (the U-shaped criterion function was not applied in the decision-making model under

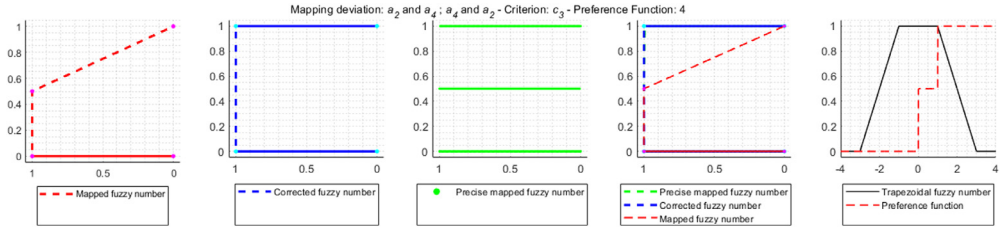

Fig. 12. Error and correction during the mapping of deviations $\widetilde{d}_3(\widetilde{a}_2, \widetilde{a}_4)$ and $\widetilde{d}_3(\widetilde{a}_4, \widetilde{a}_2)$.

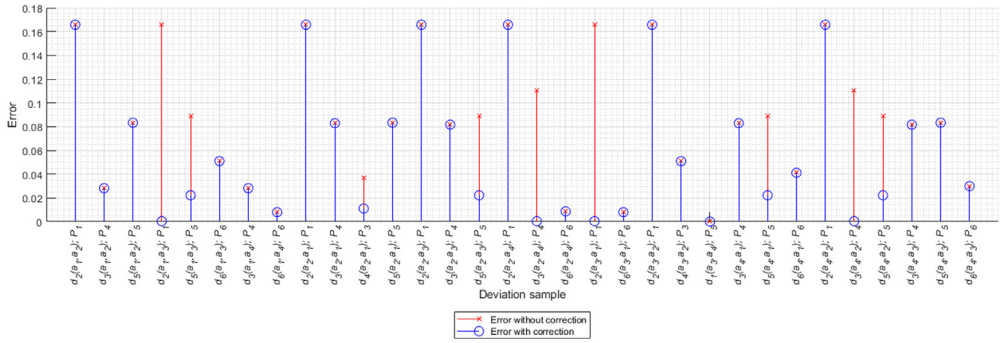

Fig. 13. Diagram of errors during mapping.

consideration).

$$Dc\left(\mu_{\widetilde{P}_k(\widetilde{d}_k)}\right) = \frac{\int_{y_{min}}^{y_{max}} y \cdot \mu_{\widetilde{P}_k(\widetilde{d}_k)}(y)dy}{\int_{y_{min}}^{y_{max}} \mu_{\widetilde{P}_k(\widetilde{d}_k)}(y)dy} \tag{29}$$

$$Dt\left(\mu_{\widetilde{P}_k(\widetilde{d}_k)}\right) = \frac{P_k(d_{k3})^2 + P_k(d_{k4})^2 + P_k(d_{k3})P_k(d_{k4}) - P_k(d_{k1})^2 - P_k(d_{k2})^2 - P_k(d_{k1})P_k(d_{k2})}{3(P_k(d_{k3}) + P_k(d_{k4}) - P_k(d_{k1}) - P_k(d_{k2}))} \tag{30}$$

$$error(P) = \sum_{k=1}^{n} \left| Dc\left(\mu_{\widetilde{P}_k(\widetilde{d}_k)}\right) - Dt\left(\mu_{\widetilde{P}_k(\widetilde{d}_k)}\right) \right| \tag{31}$$

The error values obtained during mapping with and without correction are shown in Table 5.

The results presented in Table 5 clearly show that the solution obtained by applying the correction is less error prone. This is confirmed by the diagram of errors during mapping, shown in Fig. 13. The analysis of Fig. 13 indicates that in the decision problem under consideration, if a correction is made, it reduces the mapping error in each case.

The presented analyses allow us to conclude that the decision problem solution obtained using the NEAT F-PROMETHEE method (with correction) has a smaller error than the solution obtained using Fuzzy PROMETHEE (without correction). Therefore, it can be concluded that the ranking of NEAT F-PROMETHEE II ($\widetilde{a}_3 \succ \widetilde{a}_2 \succ \widetilde{a}_1 \succ \widetilde{a}_4$) and the partial order shown in Fig. 6 is correct. Additionally, the presented calculation example shows that the correction made even when mapping a small number of deviations can significantly change the ranking of considered alternatives.

Conclusion

The article presents the methodological basis of the NEAT F-PROMETHEE method and the details of its implementation in MATLAB. Moreover, the calculation results of the NEAT F-PROMETHEE

method were compared with the standard Fuzzy PROMETHEE method based on TFNs. This comparison was made using the management decision-making problem, which was solved using both multi-criteria decision support methods. The results of the conducted research indicate that the NEAT F-PROMETHEE method allows to obtain more precise results, with a lower error resulting from the use of TFNs. Development of the NEAT F-PROMETHEE implementation in the MATLAB environment will increase the ease of use of this method. This will allow users to focus on better modelling of the decision problems under consideration, instead of worrying about the details related to the correct implementation of the method. As for further directions of research on the NEAT F-PROMETHEE method, in the context of sustainable management, it seems interesting to combine this method with the PROSA method [11,12]. This would allow uncertainty and imprecision to be taken into account in the decision-making problems of sustainable development, where the balance between economic, social and environmental factors is important. Yet another interesting research challenge is the development of GAIA (Geometrical Analysis for Interactive Assistance) [23] for NEAT F-PROMETHEE using TFNs. It would allow analysing the fuzzy decision problem from a descriptive perspective.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Science Centre, Poland, Grant no. 2019/35/D/HS4/02466.

References

[1] P. Ziemba, Multi-criteria approach to stochastic and fuzzy uncertainty in the selection of electric vehicles with high social acceptance, expert systems with applications, Expert Syst. Appl. 173 (2021) 114686, doi:10.1016/j.eswa.2021.114686.
[2] J.P. Brans, Y. De Smet, PROMETHEE methods, in: S. Greco, M. Ehrgott, J.R. Figueira (Eds.), Multiple Criteria Decision Analysis: State of the Art Surveys, Springer, New York, NY, 2016, pp. 187–219, doi:10.1007/978-1-4939-3094-4_6.
[3] J. Geldermann, T. Spengler, O. Rentz, Fuzzy outranking for environmental assessment. Case study: iron and steel making industry, Fuzzy Sets Syst. 115 (2000) 45–65, doi:10.1016/S0165-0114(99)00021-4.
[4] P.F. Drucker, The Practice of Management, Harper Collins, 2010.
[5] M. Puseljic, A. Skledar, I. Pokupec, Decision-making as a management function, Interdiscip. Manag. Res. 11 (2015) 234–244.
[6] M. Taghavifard, K. Khalili-Damghani, R. Tavakkoli-Moghaddam, Decision making under uncertain and risky situations, in: Proceedings of the Enterprise Risk Management Symposium, 2009.
[7] S.J. Chen, C.L. Hwang, Fuzzy sets and their operations, in: S.J. Chen, C.L. Hwang (Eds.), Fuzzy Multiple Attribute Decision Making: Methods and Applications, Springer, Berlin Heidelberg, Berlin, Heidelberg, 1992, pp. 42–100, doi:10.1007/978-3-642-46768-4_3.
[8] A. Mardani, A. Jusoh, E.K. Zavadskas, Fuzzy multiple criteria decision-making techniques and applications – two decades review from 1994 to 2014, Expert Syst. Appl. 42 (2015) 4126–4148, doi:10.1016/j.eswa.2015.01.003.
[9] M. Behzadian, R.B. Kazemzadeh, A. Albadvi, M. Aghdasi, PROMETHEE: a comprehensive literature review on methodologies and applications, Eur. J. Oper. Res. 200 (2010) 198–215, doi:10.1016/j.ejor.2009.01.021.
[10] P. Ziemba, NEAT F-PROMETHEE – a new fuzzy multiple criteria decision making method based on the adjustment of mapping trapezoidal fuzzy numbers, Expert Syst. Appl. 110 (2018) 363–380, doi:10.1016/j.eswa.2018.06.008.
[11] P. Ziemba, Multi-criteria stochastic selection of electric vehicles for the sustainable development of local government and state administration units in Poland, Energies 13 (2020) 6299, doi:10.3390/en13236299.
[12] P. Ziemba, Towards strong sustainability management—a generalized PROSA method, Sustainability. 11 (2019) 1555, doi:10.3390/su11061555.
[13] P. Ziemba, Inter-criteria dependencies-based decision support in the sustainable wind energy management, Energies 12 (2019) 749, doi:10.3390/en12040749.
[14] P. Ziemba, J. Becker, Analysis of the digital divide using fuzzy forecasting, Symmetry 11 (2019) 166, doi:10.3390/sym11020166.
[15] M. Kannchen, P. Ziemba, M. Borawski, Use of the PVM method computed in vector space of increments in decision aiding related to urban development, Symmetry 11 (2019) 446, doi:10.3390/sym11040446.
[16] N. Yalçın, N. Yapıcı Pehlivan, Application of the fuzzy CODAS method based on fuzzy envelopes for hesitant fuzzy linguistic term sets: a case study on a personnel selection problem, Symmetry 11 (2019) 493, doi:10.3390/sym11040493.
[17] P. Ziemba, Multi-criteria fuzzy evaluation of the planned offshore wind farm investments in Poland, Energies 14 (2021) 978, doi:10.3390/en14040978.
[18] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (1965) 338–353, doi:10.1016/S0019-9958(65)90241-X.
[19] I. Perfilieva, in: Fuzzy Function: Theoretical and Practical Point of View, Atlantis Press, 2011, pp. 480–486, doi:10.2991/eusflat.2011.142.

[20] G. van Huylenbroeck, The conflict analysis method: bridging the gap between ELECTRE, PROMETHEE and ORESTE, Eur. J. Oper. Res. 82 (1995) 490–502, doi:10.1016/0377-2217(95)98195-6.
[21] T. Holy, Generate Maximally Perceptually-Distinct Colors, MATLAB Central File Exchange, 2019 https://www.mathworks.com/matlabcentral/fileexchange/29702-generate-maximally-perceptually-distinct-colors (accessed December 17, 2019).
[22] K. Kearney, line2arrow.m, GitHub. (2019). https://www.github.com/kakearney/line2arrow-pkg (accessed December 17, 2019).
[23] B. Mareschal, J.P. Brans, Geometrical representations for MCDA, Eur. J. Oper. Res. 34 (1988) 69–77, doi:10.1016/0377-2217(88)90456-0.