pymcdm—The universal library for solving multi-criteria decision-making problems

Article · May 2023

DOI: 10.1016/j.softx.2023.101368

3 authors:

Bartłomiej Kizielewicz
West Pomeranian University of Technology, Szczecin
60 PUBLICATIONS **552** CITATIONS
SEE PROFILE

Andrii Shekhovtsov
West Pomeranian University of Technology, Szczecin
52 PUBLICATIONS **662** CITATIONS
SEE PROFILE

Wojciech Sałabun
West Pomeranian University of Technology, Szczecin
150 PUBLICATIONS **2,905** CITATIONS
SEE PROFILE

SoftwareX 22 (2023) 101368

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

pymcdm—The universal library for solving multi-criteria decision-making problems

Bartłomiej Kizielewicz [a], Andrii Shekhovtsov [b], Wojciech Sałabun [a,*]

[a] West Pomeranian University of Technology in Szczecin, ul. Żołnierska 49, Szczecin, 71-210, Poland
[b] National Telecommunications Institute, ul. Szachowa 1, Warsaw, 04-894, Poland

ARTICLE INFO

Article history:
Received 16 September 2022
Received in revised form 8 March 2023
Accepted 10 March 2023

Keywords:
Python
MCDA
MCDM
Decision support

ABSTRACT

Multi-criteria decision-making/analysis is a vast field designed for solving decision-making problems. Due to the widespread use of many relevant MCDA/MCDM approaches and their frequent appearance in scientific papers, there is a particular gap regarding comprehensive software that should have relevant components. Therefore, this paper proposes a flexible library written in Python 3 for multi-criteria analysis/decision-making. It includes tools related to evaluating alternative options, determination of Pareto optimal solutions, determination of criteria relevance, comparative analysis, and visualization. The novelty that distinguishes the library from other software is the provision of a wide range of approaches, high performance, consistency of the software, and its small size. The research presented in this paper presents its high potential as a tool that decision-makers and domain experts can use in the decision-making process.

Code metadata

Current code version	1.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00263
Legal Code License	MIT Licence
Code versioning system used	Git
Software code languages, tools, and services used	Python >= 3.8, numpy==1.22.3, matplotlib >= 3.6.0rc2, scipy==1.9.1
Compilation requirements, operating environments & dependencies	Python v3.8
Link to developer documentation	https://pymcdm.readthedocs.io/en/master/
Support email for questions	A.Shekhovtsov@zut.edu.pl, bartlomiej-kizielewicz@zut.edu.pl, wojciech.salabun@zut.edu.pl

1. Motivation and significance

Multi-criteria decision support and multi-criteria decision analysis is a field that deals with solving complex decision-making problems [1]. These problems arise in many important areas of life [2], so newer and newer Multi-criteria decision analysis/Multi-criteria decision making (MCDA/MCDM) methods are being developed to solve them [3–5]. The occurrence of conflicting criteria in decision-making problems is one of the difficulties encountered. Mainly two types of criteria are distinguished, namely the profit criterion, which says that a higher value of the criterion is the most desirable, and the cost criterion, which says that a lower value of the criterion is the most desirable. Many sources also consider the third criterion, a non-linear criterion with a different/other than the most minor and most significant value of the most desirable criterion [6,7]. Therefore, simple techniques may not be enough when evaluating decision-making options.

One of the classic MCDA/MCDM methods capable of solving decision problems with different criteria is the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) approach. This approach evaluates decision alternatives by considering the distance from ideal solutions [8,9]. Due to its high popularity and accuracy, many new methods are based on the concept of ideal solutions such as Evaluation based on Distance from Average Solution (EDAS), Stable Preference Ordering Towards Ideal Solution (SPOTIS), Measurement of Alternatives and Ranking according to COmpromise Solution (MARCOS), COmplex

* Corresponding author.
E-mail address: wojciech.salabun@zut.edu.pl (Wojciech Sałabun).

https://doi.org/10.1016/j.softx.2023.101368

PRoportional ASsessment (COPRAS), or COmbinative Distance-based ASsessment (CODAS) [10–12]. Another classic technique is the VIšekriterijumsko KOmpromisno Rangiranje (VIKOR) compromise approach, which evaluates alternatives by seeking to compromise designated vector preferences [13,14]. Also, one of the methods that use compromise in its operation is the COmbined COmpromise SOlution (COCOSO) method, which combines the operation of the Weight Sum Model (WSM) and Weighted Product Model (WPM) methods [15]. Another interesting method is Ranking of Alternatives through Functional mapping of criterion sub-intervals into a Single Interval (RAFSI) which is used functional mapping in order to order alternatives [16]. In addition to approaches based on trade-offs or distances from ideal solutions, there are other approaches, such as Characteristic Objects METhod (COMET), a method based on evaluating alternatives using fuzzy rules [17]. There are also families of approaches Preference Ranking Organization Method for Enrichment of Evaluations (PROMETHEE) and ELimination Et Choix Traduisant la REalité - ELimination and Choice Expressing the REality (ELECTRE) based on evaluating alternatives using their comparisons.

It is not easy to put their significance together when multiple criteria occur quickly. Expressing the relevance of criteria using a vector of weights is sometimes problematic and often unintuitive to the decision maker/expert. With existing MCDA/MCDM solutions, it is possible to extract the knowledge of the decision maker/expert simply, for example, using pairwise comparisons. In addition, there are also solutions using objective measures relating to information that can automatically adapt to a given decision problem to determine the relevance of criteria.

Among the classic methods for objective criteria selection, weights are the Entropy method, which is based on the measure of information proposed by Shannon [18]. Also, classical methods for determining the significance of popular criteria in decision-making processes are based on standard deviation and variance. In the main, these methods are centered around measures that determine specific information about criteria. The Criteria Importance Through Inter-criteria Correlation (CRITIC) method determines the significance of criteria based on their correlation [19]. The Method Based on the Removal Effects of Criteria (MEREC) on the other hand, determines criterion weights based on their exclusion and evaluation with a specific function based on logarithm [20].

Given their flexibility and high accuracy, these methods are popular and used in management, sustainable transportation, renewable energy, and medicine. Unfortunately, the functionality of many novel methods is occasionally challenging to find. In addition, it is mainly provided related works in the form of libraries for complex languages or commercial languages such as Julia, R, MATLAB, or Java [21,22]. The continuous development of multi-criteria decision-making/analysis methods also creates a gap related to the lack of tools to use them. In addition, existing tools with slightly older methods are not developed and updated and thus less available to current standards.

The Python language is a simple tool for implementing software that has been growing tremendously over the past few years. Its capabilities include library machine learning, computer vision, signal processing, and mathematical calculations. These libraries are clear and readable, and their implementations are mainly based on standard modules. Unfortunately, existing implementations of libraries related to MCDA/MCDM topics have an unreadable structure, limiting their adaptability [23,24]. In addition, they contain numerous libraries with similar functions and increase the volume of the entire software. They also use Python language standards inefficiently, making them inefficient.

2. Contribution

This paper focuses on an open source proposal for a library called `pymcdm`, which is designed to address decision-making using MCDA/MCDM approaches. This library is written in Python 3 and includes many classic and novel approaches employed in decision-making. Also, the library features a transparent and consistent framework and uses a limited number of libraries associated with functions, due to which high performance has been provided. In addition, the library is available through the Python Package Index (PyPi), which makes it easy to install and update. Moreover, it is also published on the GitLab repository under the MIT license, making it even more accessible.

The `pymcdm` library is designed and developed to provide a significant degree of flexibility for the user. In addition, written documentation describes all functions in software and mathematical contexts, giving the user the knowledge to determine that he is choosing to employ the right tool. Moreover, projected tests based on sources from the literature ensure the reliability of the library and the outputs it provides.

3. Software description

The `pymcdm` library is designed to solve decision-making problems with the characteristics of evaluating alternatives, determining the significance of criteria, sensitivity analysis, similarity analysis, and determining Pareto-optimal solutions. The library mainly builds on the functionality of the Numpy library and Matplotlib and also uses some of the scipy libraries. The Numpy library is used for mathematical calculations in decision-making and data representation using ndarray objects. The Matplolib library, on the other hand, is used for data visualization, in which axes objects play a significant role.

In our work, we focus on the version of library 1.1.0, which is a mature and reliable proposal. `pymcdm` has undergone many changes to improve its quality, expand its core modules and add new functionality. Comparing the library in version 1.0.2, it had only 6 MCDA/MCDM methods, three weight selection approaches, and five normalization approaches. In contrast, version 1.1.0 has 15 MCDA/MCDM methods, ten weight selection methods, and eight normalization approaches. This assures the progressive work on the library and taking care of its content. In addition, the library has undergone many complete and extensive updates, which are presented using a Table 1.

3.1. Software architecture

The `pymcdm` library has many modules designed for different types of MCDA/MCDM problems. The `methods` subpackage contains MCDA/MCDM methods that evaluate alternatives presented in the form of a decision matrix. The `weights` submodule is responsible for weight selection methods associated with decision-making problems that determine the relevance of decision criteria. The `normalizations` submodule is responsible for functions related to the normalization of criteria so that they can be considered together in the decision-making process regardless of their actual ranges. The `correlations` submodule is responsible for correlation/similarity coefficients, with the help of which it is possible to determine the similarities of preferences/rankings obtained from the methods. The `helpers` submodule is the module for determining rankings, normalization matrices, and correlation matrices. The last `visuals` subpackage provides functionality with which it is possible to present COMET and PROMETHEE models and rankings, preferences, and weights in graphical form. The content of the subpackages/submodules can be presented as follows:

Table 1
State of Art `pymcdm` Python library.

Version	Update date	Description software update	Added content
1.0.2	29 January 2021	Stable version of library	• MCDA/MCDM methods: COMET, COPRAS, PROMETHEE II, SPOTIS, TOPSIS, VIKOR • Weighing methods: Equal, Entropy, Standard deviation • Normalization approaches: Min–max, Max, Sum, Vector, Log • Correlation coefficients: r_s, r_w, WS, ρ, γ, τ
1.0.3	17 February 2021	Improvement and validation	• Implement an improved version of the COMET method • Validation of input data in meotds: VIKOR, COPRAS and SPOTIS
1.0.4	15 August 2021	Expansion update	• MCDA/MCDM methods: ARAS, COCOSO, CODAS, EDAS, MABAC, MARCOS, OCRA, MOORA • Weighing methods: MEREC, CRITIC, CILOS, IDOCRIW,Angle, Gini, Variance • Normalization approaches: Linear, Nonlinear, Enhanced accuracy
1.1.0	15 September 2022	Documentation, testing and visualization	• Documentation: Examples, API descriptions and method references • Tests: MCDA/MCDM methods, weighting methods, normalization approaches • Visualization: visualizations of rankings, weights, PROMETHEE and COMET

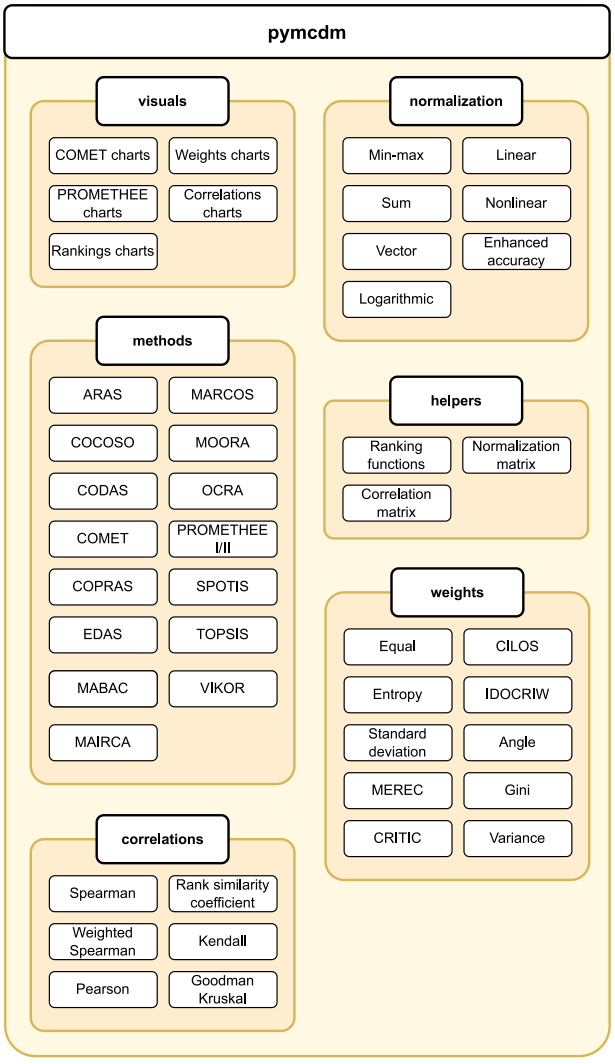

Fig. 1. Framework of the `pymcdm` library.

- `methods`—subpackage of 15 MCDA/MCDM evaluation methods: (1) ARAS, (2) COCOSO, (3) CODAS, (4) COMET, (5) COPRAS, (6) EDAS, (7) MABAC, (8) MAIRCA, (9) MARCOS, (10) MOORA, (11) OCRA, (12) PROMETHEE, (13) SPOTIS, (14) TOPSIS, (15) VIKOR;
- `weights`—submodule of 10 MCDA/ MCDM methods to determine the relevance of criteria: (1) Equal weights, (2) Entropy weights, (3) Standard Deviation weights, (4) MEREC, (5) CRITIC, (6) CILOS, (7) IDOCRIW, (8) Angle weights, (9) Gini weights, (10) Variance weights;
- `normalizations`—submodule 8 normalization functions by profit/cost criteria: (1) Min–max, (2) Max, (3) Sum, (4) Vector, (5) Logarithmic, (6) Linear, (7) Nonlinear, (8) Enhanced accuracy;
- `correlations`—submodule of 6 correlation/similarity coefficients: (1) Spearman's rank correlation coefficient, (2) Pearson correlation coefficient, (3) Weighted Spearman's rank correlation coefficient, (4) Rank similarity coefficient, (5) Kendall rank correlation coefficient, (6) Goodman and Kruskal's Gamma Coefficient;
- `visuals`—subpackage of 13 functions for visualizing important MCDA/MCDM data: (1) `ranking_scatter`, (2) `ranking_flows`, (3) `polar_plot`, (4) `polar_weights`, (5) `weights_plot`, (6) `mej_2d_plot`, (7) `mej_3d_plot`, (8) `promethee_I_flows`, (9) `promethee_I_graph`, (10) `rankings_bar`, (11) `correlation_heatmap`;
- `helpers`—submodule for help functions: (1) `rankdata`, (2) `rrankdata`, (3) `correlation_matrix`, (4) `normalize_matrix`.

Considering the library pymcdm subpackages and submodules the structure can be presented graphically in Fig. 1.

3.2. Software functionalities

The `pymcdm` library has several useful functionalities for making multi-criteria decisions. The library's functionalities revolve around evaluating decision options and determining the relevance of criteria and the desired information for decision-makers. The implemented algorithms provide a simple output form and intuitive interface for user input. The main functionalities of the library can be described as follows:

- MCDA/MCDM methods provide functionality related to the evaluation of decision options;
- Approaches related to the selection of criteria weights provide functionality related to objectively determining the relevance of attributes using measures of information;
- New MCDA/MCDM approaches resistant to the paradox of reversal of rankings (SPOTIS, COMET);
- Providing input normalization approaches that are needed to ensure the reliability of the results obtained;
- Providing similarity/correlation coefficients through which it is possible to determine the relationship between rankings, preferences, and weights of decision problems;
- Providing parametric MCDA/MCDM compromise methods such as VIKOR and COCOSO;
- Providing functionality related to visualization of the obtained results;
- The condition of functionality related to visualization of methods such as COMET and PROMETHEE enabling better representation of the method model and its deeper analysis;
- Methods enabling the construction of partial ranking such as PROMETHEE I.

3.3. Sample code snippets analysis

With the help of Listing 1, an example code snippet written in Python 3 using the functionalities of the `pymcdm` library is presented. The presented example will use the `TOPSIS` class included in the `methods` subpackage and the `rrankdata` function included in the `helpers` subpackage. In order to use the algorithms included in the library, they must be imported, as shown in lines 2–3 of the following listing. After importing the necessary methods, the user defines the input data. In the case of the TOPSIS method, the input data are a decision matrix, a vector of criteria weights, and a vector of criteria types. The container that holds the data is a `ndarray` derived from the `Numpy` library. In order to define a decision matrix having two criteria and four alternatives, a list structure was used. This list was given as input to the `array` function, thanks to creating an instance of the class. In addition, the parameter given is the array type, which was set to float due to floating point operations. The effect of the above steps is shown in lines 6–11. After the creation of the decision matrix, the creation of the vector of weights in line 14 follows. Due to the limitations of methods, the sum of values of the vector of weights should be equal to 1. Therefore, to establish the same relevance of the two criteria, the weights were set with a value of 0.5. Once the vector of weights is determined, it is followed by establishing the types of criteria present in the matrix using line 15 in the following listing. The vector of criteria types can consist of a value of −1 corresponding to a cost-type criterion and a value of 1 corresponding to a profit-type criterion. The criteria types and weights are assigned in the same order as the decision matrix.

Listing 1: Example of using the `pymcdm` library

```
1  import numpy as np
2  from pymcdm.methods import TOPSIS
3  from pymcdm.helpers import rrankdata
4
5  # Define decision matrix (2 criteria, 4
       alternatives)
6  alts = np.array([
7      [4, 4],
8      [1, 5],
9      [3, 2],
10     [4, 2]
11 ], dtype='float')
12
13 # Define weights and types
14 weights = np.array([0.5, 0.5])
15 types = np.array([1, -1])
16
17 # Create object of the method
18 topsis = TOPSIS()
19
20 # Determine preferences and ranking for
       alternatives
21 pref = topsis(alts, weights, types)
22 ranking = rrankdata(pref)
23
24 # Alternatively:
25 ranking = topsis.rank(pref)
26
27 for r, p in zip(ranking, pref):
28     print(r, p)
```

After defining all the inputs of the method, create its object when calling the imported class. In the case of the TOPSIS method, parameters are not required. However, arguments may be required for other methods, so this should be considered when creating the object. The process of creating the object is shown in line 18. Then it would help if the created object were used to evaluate the alternatives included in the decision matrix. To do this, we call the object a function to which we pass the decision matrix, criterion weights, and criterion types. After calling the object, the preferences of the alternatives are returned in the form of a `ndarray` determined by the TOPSIS method, which in this listing are saved to the `pref` variable in line 21. To determine the ranking, the `rrankdata` function is used to pass the preferences of the alternatives from the TOPSIS method as an argument, as shown in line 22. Alternatively, we can use `rank` method that is included in every MCDA object class, which will order preferences in correct way. After determining the preferences and ranking, they are displayed by iterating over successive values of the `ranking` and `pref` variables in lines 24–25.

Table 2
Selected criteria for the problem of evaluating vans.

C_i	Name	Unit	Type
C_1	Carryfying capacity	[kg]	Profit
C_2	Max velocity	[km/h]	Profit
C_3	Travel range	[km]	Profit
C_4	Engine power	[kW]	Profit
C_5	Engine torque	[Nm]	Profit
C_6	Battery charging 100%	[h]	Cost
C_7	Battery charging 80%	[min]	Cost
C_8	Battery capacity	[kWh]	Profit
C_9	Price	[thous. USD]	Cost

4. Illustrative example

This section will provide brief illustrated examples of using the `pymcdm` library. For the example application of the library, the problem of evaluation of electronic vans was chosen. Therefore, 9 criteria components were defined, whose names, along with data such as units and types of criteria, are presented using a Table 2. For the study, the 10 electronic vans were used and presented in the form of a decision matrix using a Table 3. All the dataset is included in the `examples` in the `pymcdm` library.

Four MCDA/MCDM methods, i.e., TOPSIS, MABAC, COMET, and SPOTIS, were arbitrarily selected to present the evaluation of alternative options. Due to the input data requirements of the methods, weights were determined using an objective method based on the standard deviation measure. For the COMET method, 3 characteristic values [min, mean, max] were selected for each criterion based on the decision matrix. In contrast, for the SPOTIS approach, [min, max] values derived from the decision matrix were defined as the boundary values for the criteria. The TOPSIS method was also used for the COMET approach to evaluate characteristic objects to create the MEJ matrix. The described

Table 3
Decision matrix consisting of A_1–A_{10} alternatives for the example of van evaluation.

A_i	Name	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
A_1	EVI MD	3000	96	145	200	610	10.0	120	99.0	120.0
A_2	EVI Walk-In Van	2000	100	145	200	610	10.0	120	99.0	90.0
A_3	e-NV200+	705	120	170	80	270	4.0	30	24.0	25.0
A_4	e-Wolf Omega 0.7	613	140	180	140	400	8.0	40	24.2	50.0
A_5	Minicab-MiEV Truck	350	100	110	30	196	4.5	15	10.5	12.9
A_6	Mitsubishi Minicab-MiEV (10.5 kWh)	350	100	100	30	196	4.5	15	10.5	15.5
A_7	Mitsubishi Minicab-MiEV (16 kWh)	350	100	150	30	196	7.0	35	16.0	18.7
A_8	Partner Panel Van	635	110	170	49	200	8.0	35	22.5	31.5
A_9	Phoenix Motorcars SUV	340	150	160	110	500	6.0	10	35.0	45.0
A_{10}	Piaggio Porter electric-power	750	57	110	10	80	8.0	120	35.0	24.4

Table 4
Evaluations of alternatives obtained from the considered MCDA/MCDM approaches.

A_i	TOPSIS	MABAC	COMET	SPOTIS
A_1	0.937	0.647	0.949	0.084
A_2	0.636	0.403	0.716	0.328
A_3	0.165	−0.015	0.232	0.746
A_4	0.170	0.011	0.279	0.720
A_5	0.074	−0.155	0.099	0.886
A_6	0.074	−0.159	0.098	0.890
A_7	0.071	−0.152	0.093	0.883
A_8	0.131	−0.067	0.172	0.798
A_9	0.159	−0.028	0.249	0.759
A_{10}	0.153	−0.131	0.109	0.862

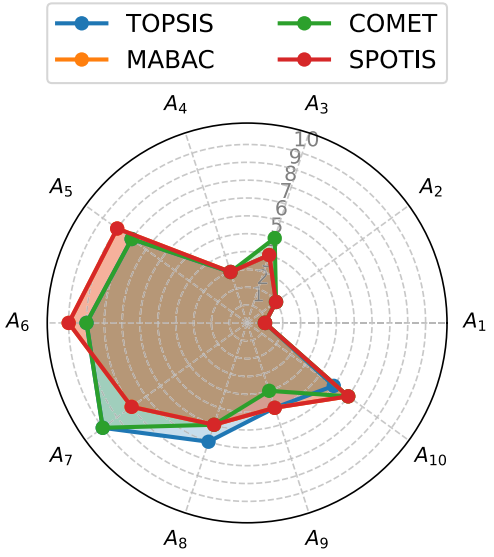

Fig. 2. Graphical representation of the rankings using the `polar_plot` function.

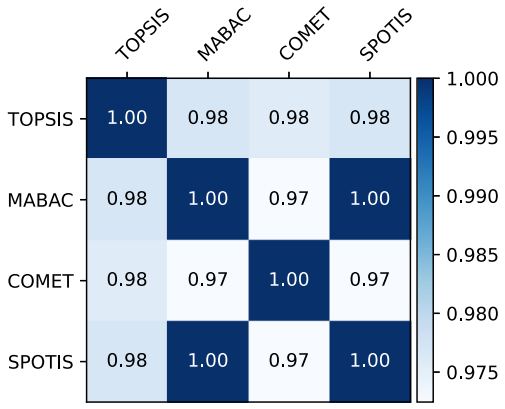

Fig. 3. Example correlation matrix for the obtained rankings.

functionality is implemented in the library, and all the code is presented in examples. The determined evaluations for each method are shown using Table 4.

Rankings were calculated based on the determined ratings of individual vans derived from the decision matrix. They were calculated using the `rank` method derived from the MCDA/MCDM method classes. The library assigned a ranking function for each MCDA/MCDM class according to whether it ranked from the smallest or the most significant value. The results obtained were then presented in graphical form using the `polar_plot` function. Fig. 2 represents the result of this function.

In order to compare the obtained rankings, the similarity coefficients of rankings can be used, which express to what degree rankings are similar to each other. The `pymcdm` library offers such functionality via the `correlations` submodule. Due to a large number of combinations of comparisons between rankings, it is possible to use the `correlation_matrix` function, which will return a matrix of the values of a given coefficient for the rankings compared with each other in the output. In addition, for a simple graphical representation of the obtained correlations, the function `correlation_heatmap` can be used. An example of the result of using this function for the obtained rankings from TOPSIS, MABAC, COMET, and SPOTIS methods is shown in Fig. 3.

Although, in the above example, the weights were selected using an approach based on a Standard Deviation measure, it is also possible to use other methods. The weights submodule has many objective methods for selecting weights based on measures such as Entropy or the Gini index. Depending on the approach to determine the relevance of criteria in the form of weights, they may differ substantially from each other. Therefore, with the help of Fig. 4, the weights for the criteria obtained from the approaches of determining the weights of Equal, Entropy, Standard Deviation, and Gini index are presented. The determined weights were for the decision matrix used in the previous example.

In addition to visualization related to rankings, the similarity of rankings and weights using the `pymcdm` library, method models can also become visualized. One of the models that can be visualized is the COMET model. The main functionality of the COMET method is based on the MEJ (Matrix of Expert Judgment). The matrix is created by comparing characteristic objects with each other, where it is assigned values [0, 0.5, 1] depending on the preference of the decision maker/expert. Also, be created this matrix can use a method that evaluates characteristic objects, such as TOPSIS. Due to the more straightforward matrix representation, the `mej_plot` function allowed the MEJ matrix to graphical form. An example MEJ matrix is shown for criteria C_4 and C_5 at three characteristic values in Fig. 5. In addition, it is possible to visualize the COMET model for two and three criteria along with alternatives. This functionality is offered by the functions `comet_2d_plot` and `comet_3d_plot`, which take the characteristic values for criteria and alternatives

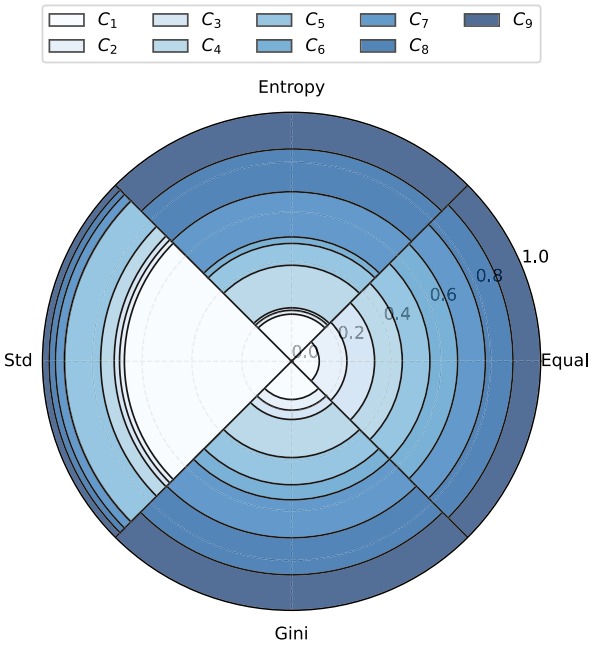

Fig. 4. Examples of determined weights for the decision matrix.

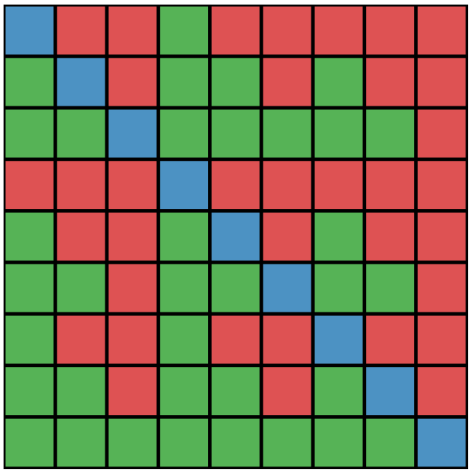

Fig. 5. Example visualization of the MEJ matrix.

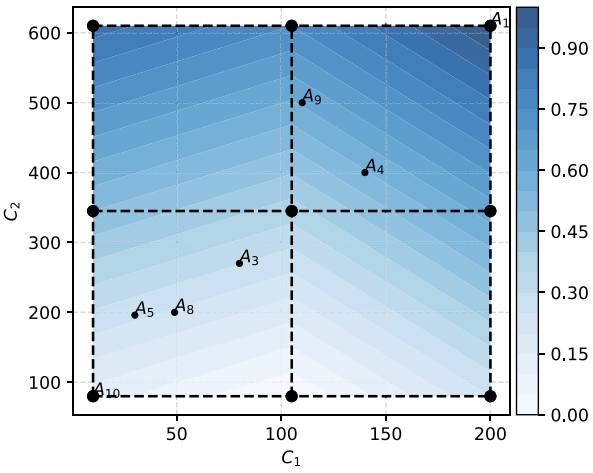

Fig. 6. Example COMET model for two criteria.

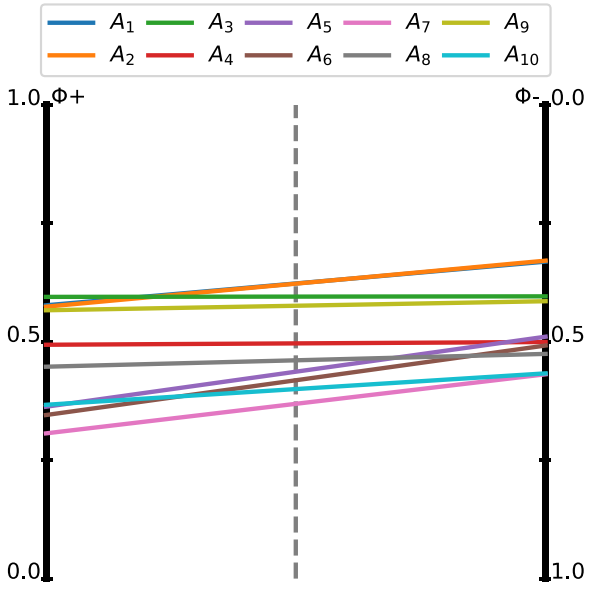

Fig. 7. Example of leaving and entering outranking flows.

as arguments. Moreover, a `comet_contourf` function based on the `comet_2d_plot` function was implemented to represent the decision-maker's preference model. In this example, the form of `comet_contourf` is presented using Fig. 6. As in the case of the MEJ matrix for this model, criteria C_4 and C_5 for three characteristic values were selected.

Besides visualizing the COMET model in the `pymcdm` library, it is also possible to visualize the PROMETHEE I model. In the main, the PROMETHEE I method is based on preference functions identified by the expert/decider. In this example, we will focus on the default function set in the library, i.e., the usual criterion. Once the function is defined, the preferences are aggregated, and outranking flows are calculated. The resulting leaving and entering outranking flows can be presented graphically using the `promethee_I_flows` function. An example of visualizing the flows is shown in Fig. 7.

In the case the PROMETHEE I method is chosen, there is also a visualization of the output model using a graph. This graph provides a richer analysis of the obtained ranking, allowing the detection of challenging scenarios to compare. In order to create a graph of a relationship between the alternatives of the PROMETHEE I model, use the `promethee_I_graph` function for leaving and entering outranking flows passed. An example graph for PROMETHEE I of the example discussed above is shown in Fig. 8.

5. Impact and discussion

Many MCDA/MCDM methods are widely used in decision-making processes. Studies show a fundamental difference in the results they receive [25–27]. The tools that exist to date mostly center around a few selected approaches, sometimes insufficient for a comprehensive analysis of a given problem [28]. This library provides many implementations of MCDA/MCDM techniques written in Python 3. It provides functionality related to trade-off approaches such as COCOSO and VIKOR, which consider the answers obtained from several generated output results [29]. In addition, the library offers methods resistant to reverse rankings, such as SPOTIS and COMET, so decision-makers can create a consistent decision model [30,31]. Due to its high popularity and

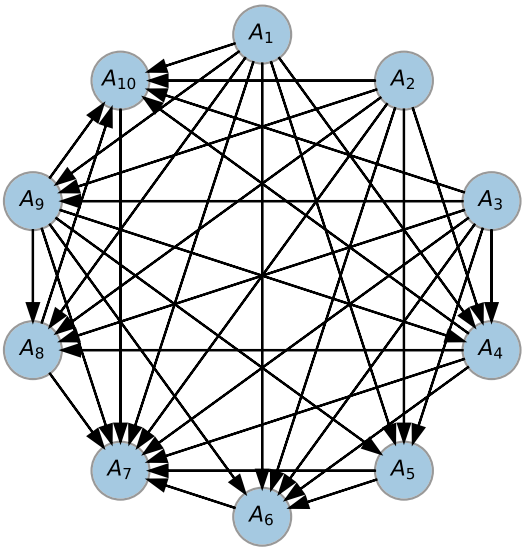

Fig. 8. Example graph for the PROMETHEE I model.

accuracy, the library offers approaches based on distance from reference points such as TOPSIS, MARCOS, and CODAS [32]. The provided functions related to MCDA/MCDM techniques allow the decision maker to examine and generally evaluate the results they receive using similarity/correlation coefficients, which are also part of the library. Sometimes it is difficult for the decision-maker to determine to what extent the criteria are relevant to the problem at hand, either under their complexity or the expertise required [33]. In addition, the representation of the relevance of the criteria in the form of a vector of weights can often be counter-intuitive for the decision-maker, making it difficult to work out. However, there are many approaches based on objective measures of information that can determine a vector of weights. The pymcdm library provides many methods for selecting objective weights using measures such as entropy, standard deviation, variance, or Gini index. These approaches are widely used in decision-making problems, and a high degree of efficiency in solving decision-making problems is achieved through them. In addition, thanks to them, many different scenarios can be studied without involving the decision maker in the iterative process of creating a decision model [34]. The library also offers functionality related to the visualization of results. With it, the user can easily present the input/output data obtained and the decision models created.

A novelty in the library we offer, which is much needed in the MCDA/MCDM software engineering community, is the straightforward and intuitive implementation of functions related to decision-making. Many `Python 3` programmers will feel free to navigate the library due to its flexible structure. In addition, the library's MIT license sues the use of written implementations by the community, thus ensuring software development and the ability to implement it in various projects. The consistent name of the library ensures that it is easy to find. Under the library's placement in, `PyPi` can be installed without difficulty. Many MCDA/MCDM libraries such as `pyrepo` provide similar functionality to modules such as `seaborn`, `pandas`, and `matplotlib`, increasing the volume of the library itself [24]. Our proposed approach centers around a single library related to visualization, i.e., `matplotlib`. In addition, in contrast to the `pyrepo` library, functionality is used to create and operate on `axes` objects. Thus, the user has much more freedom to visualize the results obtained. In addition, the `pymcdm` library uses the functionality of numerical libraries, thus ensuring high performance.

6. Conclusions

This paper presents the pymcdm library written in Python 3 for solving decision-making problems. This library has various functions, such as evaluating alternatives, determining the significance of criteria, normalizing, determining similarity/correlation of rankings/preferences, and visualizing the obtained results. Among existing libraries, it stands out for its simplicity, volume, and efficiency, making it an adequate MCDA/MCDM tool. Due to its accessibility through the `GitLab` repository and `PyPi`, it has the attributes to become one of the more widespread analyses or decision-making software. In addition, it has a wide range of methods through which the user can select the optimal settings and structure/combinations for them.

However, the software has some limitations related to the Python environment, such as code execution speed. Moreover, it is a library, so it lacks a graphical user interface (GUI), which would enhance navigating through the entire library and might attract more users who lack practice in the Python language.

Future research directions would include the development of additional modules related to subjective determination of criteria relevance, which would be directed at the interface for decision experts. Additionally, implementing algorithms providing trade-off rankings such as Copeland or Borda would need to be considered. Also, the expansion of MCDA/MCDM methods of the European school and the rule-based school would need to be considered.

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.

Data availability

No data was used for the research described in the article

Acknowledgments

The work was supported by the National Science Centre, Poland 2018/29/B/HS4/02725 and 2021/41/B/HS4/01296 (B.K. and W.S.).

References

[1] Akram M, Bibi R, Deveci M. An outranking approach with 2-tuple linguistic fermatean fuzzy sets for multi-attribute group decision-making. Eng Appl Artif Intell 2023;121:105992.
[2] Youssef MI, Webster B. A multi-criteria decision making approach to the new product development process in industry. Rep Mech Eng 2022;3(1):83–93.
[3] Biswas T, Saha P. Selection of commercially available scooters by new MCDM method. Int J Data Netw Sci 2019;3(2):137–44.
[4] Stević Ž, Pamučar D, Puška A, Chatterjee P. Sustainable supplier selection in healthcare industries using a new MCDM method: Measurement of alternatives and ranking according to COmpromise solution (MARCOS). Comput Ind Eng 2020;140:106231.
[5] Deveci M, Rodríguez RM, Labella Á, Ciftci ME. A decision support system for reducing the strategic risk in the schedule building process for network carrier airline operations. Ann Oper Res 2022;1–37.
[6] Kizielewicz B, Więckowski J, Paradowski B, Sałabun W. Dealing with non-monotonic criteria in decision-making problems using fuzzy normalization. In: International conference on intelligent and fuzzy systems. Springer; 2022, p. 27–35.
[7] Aydin N, Seker S, Deveci M, Ding W, Delen D. A linear programming-based QFD methodology under fuzzy environment to develop sustainable policies in apparel retailing industry. J Clean Prod 2023;135887.
[8] Heydari A, Niroomand S, Garg H. An improved weighted principal component analysis integrated with TOPSIS approach for global financial development ranking problem of Middle East countries. Concurr Comput: Pract Exper 2022;34(13):e6923.

[9] Garg H, Ali Z, Mahmood T, Ali MR. TOPSIS-method based on generalized dice similarity measures with hamy mean operators and its application to decision-making process. Alexand Eng J 2023;65:383–97.
[10] Ecer F, Pamucar D. MARCOS technique under intuitionistic fuzzy environment for determining the COVID-19 pandemic performance of insurance companies in terms of healthcare services. Appl Soft Comput 2021;104:107199.
[11] Krishankumar R, Pamucar D, Deveci M, Ravichandran KS. Prioritization of zero-carbon measures for sustainable urban mobility using integrated double hierarchy decision framework and EDAS approach. Sci Total Environ 2021;797:149068.
[12] Torkayesh AE, Deveci M, Karagoz S, Antucheviciene J. A state-of-the-art survey of evaluation based on distance from average solution (EDAS): Developments and applications. Expert Syst Appl 2023;119724.
[13] Zolfani S, Yazdani M, Pamucar D, Zarate P. A VIKOR and TOPSIS focused reanalysis of the MADM methods based on logarithmic normalization. 2020, arXiv preprint arXiv:2006.08150.
[14] Garg H, Thanh DNH, Rizk-Allah RM. VIKOR approach for bi-level multi-criteria nonlinear fractional programming problems: new insights. Kybernetes 2022;(ahead-of-print).
[15] Yazdani M, Wen Z, Liao H, Banaitis A, Turskis Z. A grey combined compromise solution (CoCoSo-G) method for supplier selection in construction management. J Civ Eng Manage 2019;25(8):858–74.
[16] Deveci M, Gokasar I, Pamucar D, Chen Y, Coffman D. Sustainable e-scooter parking operation in urban areas using fuzzy Dombi based RAFSI model. Sustainable Cities Soc 2023;91:104426.
[17] Kizielewicz B, Shekhovtsov A, Sałabun W. How to make decisions with uncertainty using hesitant fuzzy sets? In: International conference on intelligent and fuzzy systems. Springer; 2022, p. 763–71.
[18] Wang L, Garg H, Li N. Pythagorean fuzzy interactive hamacher power aggregation operators for assessment of express service quality with entropy weight. Soft Comput 2021;25:973–93.
[19] Krishnan AR, Kasim MM, Hamid R, Ghazali MF. A modified CRITIC method to estimate the objective weights of decision criteria. Symmetry 2021;13(6):973.
[20] Keshavarz-Ghorabaee M, Amiri M, Zavadskas EK, Turskis Z, Antucheviciene J. Determination of objective weights using a new method based on the removal effects of criteria (MEREC). Symmetry 2021;13(4), 525.
[21] Satman MH, Yıldırım BF, Kuruca E. JMcDM: A julia package for multiple-criteria decision-making tools. J Open Source Softw 2021;6(65):3430.
[22] Peters GJY. MDMA policy think tank. 2022, URL osf.io/h58r6.
[23] Cabral JB, Luczywo NA, Zanazzi JL. Scikit-criteria: Colección de métodos de análisis multi-criterio integrado al stack científico de Python. In: XLV Jornadas argentinas de informática e investigación operativa (45JAIIO)- XIV simposio argentino de investigación operativa (SIO) (Buenos Aires, 2016). 2016, p. 59–66.
[24] Wątróbski J, Bączkiewicz A, Sałabun W. Pyrepo-mcda—Reference objects based MCDA software package. SoftwareX 2022;19:101107.
[25] Petrović G, Mihajlović J, Ćojbašić Ž, Madić M, Marinković D. Comparison of three fuzzy MCDM methods for solving the supplier selection problem. Facta Universit Ser: Mech Eng 2019;17(3):455–69.
[26] Baydaş M, Pamučar D. Determining objective characteristics of MCDM methods under uncertainty: An exploration study with financial data. Mathematics 2022;10(7):1115.
[27] Ulutaş A, Balo F, Sua L, Demir E, Topal A, Jakovljević V. A new integrated grey MCDM model: Case of warehouse location selection. Facta Universit Ser: Mech Eng 2021.
[28] Yadav V, Karmakar S, Kalbar PP, Dikshit AK. PyTOPS: A python based tool for TOPSIS. SoftwareX 2019;9:217–22.
[29] Ulutaş A, Karakuş CB, Topal A. Location selection for logistics center with fuzzy SWARA and CoCoSo methods. J Intell Fuzzy Systems 2020;38(4):4693–709.
[30] Dezert J, Tchamova A, Han D, Tacnet J-M. The SPOTIS rank reversal free method for multi-criteria decision-making support. In: 2020 IEEE 23rd international conference on information fusion. IEEE; 2020, p. 1–8.
[31] Aydemir AE, Temizel TT, Temizel A, Preshlenov K, Strahinov DM. A dimension reduction approach to player rankings in European football. IEEE Access 2021;9:119503–19.
[32] Ersoy Y. Equipment selection for an e-commerce company using entropy-based TOPSIS, EDAS and CODAS methods during the COVID-19. LogForum 2021;17(3).
[33] Kizielewicz B, Paradowski B, Więckowski J, Sałabun W. Identification of weights in multi-cteria decision problems based on stochastic optimization. 2022.
[34] Kizielewicz B, Więckowski J, Shekhovtsov A, Wątróbski J, Depczyński R, Sałabun W. Study towards the time-based MCDA ranking analysis—A supplier selection case study. Facta Universit Ser: Mech Eng 2021;19(3):381–99.