Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal 2023, 17(4), 227–234
https://doi.org/10.12913/22998624/170187
ISSN 2299-8624, License CC-BY 4.0

Received: 2023.06.14
Accepted: 2023.07.15
Published: 2023.08.15

REDUCE – A Python Module for Reducing Inconsistency in Pairwise Comparison Matrices

Paweł Kuraś[1*], Dominik Strzalka[1], Bartosz Kowal[1], Jiří Mazurek[2]

[1] Department of Complex Systems, The Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, ul. MC Skłodowskiej 8, 35-036 Rzeszów, Poland

[2] School of Business Administration in Karvina, Silesian University in Opava, Univerzitní nám. 1934, Fryštát, 733 40 Karviná, Czech Republic

* Corresponding author's e-mail: p.kuras@prz.edu.pl

ABSTRACT

This paper introduces REDUCE, a Python module designed to minimize inconsistency in multiplicative pairwise comparisons (PC), a fundamental technique in Multi-Criteria Decision Making (MCDM). Pairwise comparisons are extensively used in various fields, including engineering science and numerical simulation methods, to compare different options based on a set of criteria. However, human errors in perception and judgment often lead to inconsistencies in pairwise comparison matrices (PCM). REDUCE addresses this issue by implementing several algorithms that identify and correct inaccurate data in PCMs, thereby reducing the inconsistency ratio. These algorithms do not require expert intervention, making REDUCE a valuable tool for both scientific research and small to medium enterprises that may not have access to costly commercial software or dedicated decision-making experts. The main functionality of the module is incorporating iterative algorithms for inconsistency reduction. The REDUCE library, written in Python and utilizing auxiliary libraries such as NumPy, SciPy, and SymPy, offers 21 functions categorized into data input helpers, consistency ratio (CR) reduction algorithms, PCM indexes, and support functions. Performance testing indicates that the library can efficiently handle matrices of varying sizes, particularly those ranging from 3x3 to 10x10, and its use significantly accelerates the process compared to spreadsheets, especially when dealing with large quantities of matrices. The library has already been used in several research papers and application tools, and its availability as a free resource opens up opportunities for small and medium-sized enterprises to leverage multi-criteria decision-making methods. Currently, there are no publicly available libraries for this solution. The authors believe that the proposed module may contribute to do better decision-making process in pairwise comparisons, not only for the circle of scientists but also for small and medium enterprises that usually cannot afford expensive commercial software and do not employ full-time experts in decision-making as they rely on the experience of their employees and free online resources. It should also contributes to the transition to Industry 4.0 and advances research in fields such as fuzzy logic, preference programming, and constructive consistent approximations.

Keywords: inconsistency, decision making support, pairwise comparisons, inconsistency index, inconsistency reduction.

INTRODUCTION

The historical background of pairwise comparisons (PC) is dated back to pre-historic times, when the first humans did decisions related to selection of better piece of wood for a bow or a better stone for hunting. The first known example described in the literature is the work of Catalan scholar and monk Ramon Llull written in the 13th century [1]. Then, in the 18th century this method was used again by Marie Jean Antoine Nicolas de Caritat for the elections framework [2]. In the XXth century a psychometrician L.L. Thurstone using this method proposed a Law of Comparative Judgments [3]. But the most known example of PC use is the analytic hierarchy process (AHP) proposed by Saaty [4] and together with analytic network process (ANP) it became one of the most frequently used multiple-criteria method applied in majority of decision making systems [5].

Together with PC a special problem of its (in)consistency is also well-known. It is expected that if for object A it's assumed that it is 2 times better than object B, and further, the object B is 5 times better than an object C, for the object A one can expect that it is 10 times better than C (i.e. 2 × 5 = 10). If not, the judgments in comparisons are inconsistent. Several measures of inconsistency were proposed – the first one was introduced by Saaty [4] by a special consistency index CI and the consistency ratio CR. Because inconsistency indicates that human's judgements are not fully consistent and this influences the final results of PC usage there is a need to propose some automatic methods that can not only indicate this problem but also help to remove inconsistency from existing matrices.

Since there are currently no free and publicly available libraries implementing the presented solutions (both in Python and other programming languages, as of 2023/07/14), up to now researchers dealing with the topic of pairwise comparisons have had to rely mainly on working in spreadsheets such as Microsoft Excel or LibreOffice Calc. In this paper authors would like to introduce Python package for reducing inconsistency of PC matrices according to several known and presented in literature algorithms. It is believed that this software can find many useful applications among researchers and enthusiasts who work with multi criteria decision making systems and still developed new challenges in this field.

The paper is organized as follows: in Section 2 there will be presented issues related to pairwise comparison matrices, the indices used to describe them and the problem of reducing the inconsistency of such matrices. In Section 3, the architecture of the REDUCE library is shown, together with a description of the functionality and a code analysis of the key functions that comprise it. Then, Section 4 will give real and existing user applications that use the REDUCE library, and the potential impact of such a free library on the development of small and medium-sized enterprises, along with Section 5 containing a summary of the entire paper.

PAIRWISE COMPARISONS

Let us assume that $C = \{c_1, c_2, ..., c_n\}$, $n \in N$, $n > 1$ is a non-empty, finite set of compared objects such as: criteria, alternatives, physical properties, preferences, etc. Having S as a PC scale one can denote by $a_{ij} \in S$, $a_{ij} > 0$ a relative preference (importance) of an object i over object j, where $i, j \in \{1, ..., n\}$. If $a_{ij} = 5$ this is understood as a situation where object i is 5 times more preferred, more important, than an object j.

A $n \times n$ square matrix $A = [a_{ij}]$ (1) is a pairwise comparisons matrix (PCM) and a PC matrix $A_{n\times n} = [a_{ij}]$ is reciprocal if (2) is satisfied.

$$A = \begin{bmatrix} 1 & a_{12} & \dots & a_{1n} \\ a_{21} & 1 & \dots & \dots \\ \dots & \dots & 1 & \dots \\ a_{n1} & \dots & \dots & 1 \end{bmatrix} \tag{1}$$

$$a_{ij} = \frac{1}{a_{ji}} \quad \forall i, j \in \{1, \dots, n\} \tag{2}$$

It is assumed that all PCMs are reciprocal. A pairwise comparisons method is a kind of decision making method that is directly related to priority deriving method that gives a priority vector $w = (w_1, ..., w_n)$ (also known as a vector of weights of all n compared objects) from A PC matrix. Usually the priority vector w is normalized (3).

$$\sum_{i=1}^{n} w_i = 1 \tag{3}$$

The values $w_1, ..., w_n$ of comparisons are used for ranking the best and the worst objects.

As it was mentioned in Section 1 a special problem of PC's (in)consistency should be adressed. Saaty's consistency index CI [4] of a n×n PC matrix A is given as (4) and the consistency ratio CR is defined as (5).

$$CI(A) = \frac{\lambda_{max} - n}{n - 1} \tag{4}$$

$$CR(A) = \frac{CI(A)}{RI(n)} \tag{5}$$

In Definition 3 $RI(n)$ is the most popular and is understood as the random consistency index expressed by the arithmetic mean of randomly generated PC matrices of a given order with Saaty's scale dependent on n, and λ_{max} is the largest (positive) eigenvalue of A.

However, there are also other definitions of inconsistency indexes – thanks to the analysis of the studies performed in Mazurek's work, it was also decided to implementation of important indexes: Relative Error Index [6], Golden-Wang index [7], Koczkodaj index [8], Obata index [9], Peláez-Lamata index [10], GCI and TGCI indexes [11, 12] and the Harmonic Consistency index [13] – definitions of which are provided in the publications of their authors, and their implementation can be found in the library described later in this research paper (except Obata index which exhibited stability issues during operation).

Because the main cause of inconsistency are errors in humans reality perception and judgements it is not so obvious where they could happened, there are methods and algorithms that are able to find and correct wrong data in matrix A in order to minimize the inconsistency ratio.

While there are many algorithms for this purpose, in principle there are two groups of pairwise comparison matrix inconsistency reduction algorithms - non-iterative and iterative [21]. Based on the conclusions of the research work by the team of Mazurek et al. [20], the library proposed in this paper includes algorithms from the iterative algorithm group. These are the algorithms by Cao et al. [27], Szybowski [28] and Xu and Wei [29]. Although there are at least two more known algorithms present in the paper [20], the algorithm of Kou et al. [30] was found to be unstable and problematic to work efficiently, as was the algorithm of Mazurek et al. [31], which does not converge to zero inconsistency in every case - so these algorithms were disregarded during the development of the library.

SOFTWARE DESCRIPTION

Software architecture

The reduce.py library was written using the Python language, using the aux-iliary libraries NumPy [16], SciPy [17] and SymPy [18]. The library repository is hosted at [19].

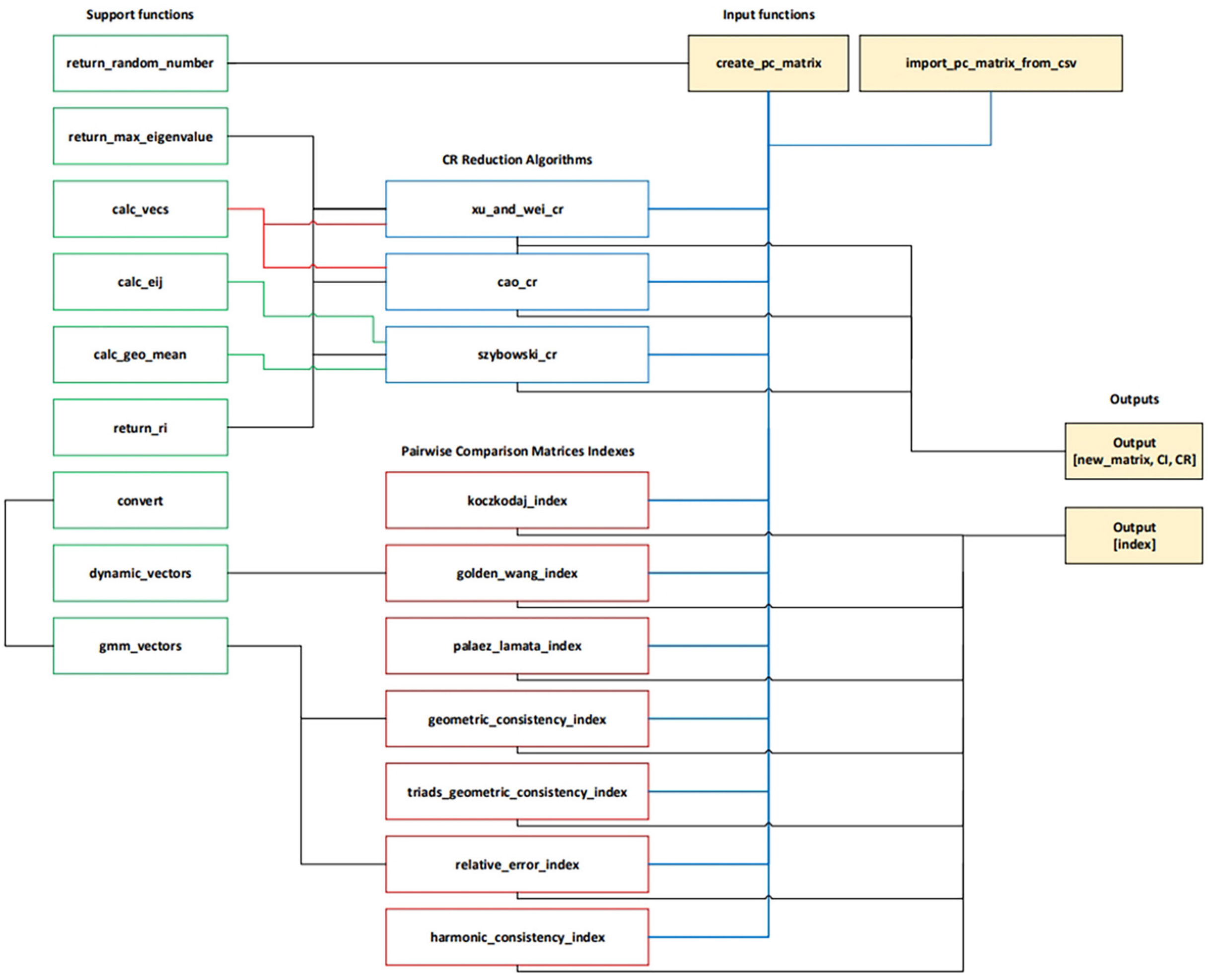

Fig. 1. Architecture of the reduce.py library

The software architecture is included in Figure 1. It consists of 21 functions in the current version. All functions are available to the end user. They can be divided into four categories: data input helpers, CR (consistency ratio) reduction algorithms, PCM (pairwise comparison matrices) indexes and support functions. The first category contains two helper functions to help both: create a random pairwise comparison matrix and perform its reading from a suitably crafted CSV file. The second category contains implementations of comparison matrix inconsistency reduction algorithms, described in the Section 2, and their implementation will be discussed later in the article. The input is given a comparison matrix with the required auxiliary parameters, while the output is a matrix with a reduced inconsistency coefficient (CR). The third category contains the indexes used in the researchers work which can be calculated by giving the pairwise comparison matrix directly on the input, and on the output receiving the given index. The last category is auxiliary functions, used by the main functions to calculate various parameters needed for the operation of the essential algorithms. Although these are functions added for the operation of the algorithms in the second and third categories, it was considered that they would be publicly available to all users, due to their potential usefulness.

Computational complexity

Upon a thorough analysis of the REDUCE library, it becomes evident that the most computationally efficient functions such as *return_random_number()* and *return_ri(size)* exhibit a constant time complexity of *O(1),* ensuring their capability to efficiently handle matrices of any size. However, it's crucial to note that these functions serve auxiliary roles within the library. The main

Table 1. REDUCE library complexity analysis

Name of the function	Complexity	Explanation
create_pc_matrix(size)	$O(n^2)$	Two nested loops each running n times
return_random_number()	$O(1)$	The operation of picking an element from a list is constant time
return_max_eigenvalue(n_matrix)	$O(n^3)$	The time complexity of the eigenvalue algorithm is cubic
calc_vecs(n_matrix)	$O(n^3)$	This complexity comes from the eigenvector calculation. The subsequent operations inside this function have lower complexity
calc_geo_mean(n_matrix)	$O(n^2)$	A single loop going over n elements, and inside the loop, another operation going over n elements
calc_eij(n_matrix)	$O(n^2)$	The operations inside the two nested loops are all constant time
return_ri(size)	$O(1)$	The function checks for certain values and returns a corresponding result
import_pc_matrix_from_csv(filename)	$O(n^2)$	This is due to reading the CSV file where n is the number of rows (or columns)
convert(imp_matrix)	$O(n^2)$	Converting each element in the n x n matrix
dynamic_vectors(n_matrix)	$O(n^3)$	This complexity is determined by the eigenvector calculation.
gmm_vectors(matrix)	$O(n^2)$	There are several loop iterations over n
xu_and_wei_cr(matrix, lambd, threshold)	$O(n^4)$	While loop that continues until a condition is met. Inside this loop, we have nested for loops iterating over the matrix, and function calls to return_max_eigenvalue and calc_vecs which are both $O(n^3)$
cao_cr(matrix, lambd, threshold)	$O(n^4)$	Similar to xu_and_wei_cr, this function also has $O(n^4)$ complexity
szybowski_cr(matrix, threshold)	$O(n^4)$	The complexity is also similar to xu_and_wei_cr
koczkodaj_index(matrix)	$O(n^3)$	Three nested loops each running n times
golden_wang_index(matrix)	$O(n^3)$	Nested loops and a function call to dynamic_vectors, which is $O(n^3)$
pelaez_lamata_index(matrix)	$O(n^3)$	Three nested loops each running n times
geometric_consistency_index(matrix)	$O(n^2)$	A couple of nested loops iterating over the matrix, and a function call to gmm_vectors which is $O(n^2)$
triads_geometric_consistency_index(matrix)	$O(n^3)$	Three nested loops each running n times.
relative_error_index(matrix)	$O(n^2)$	Two nested loops iterating over the matrix, and a function call to gmm_vectors which is $O(n^2)$
harmonic_consistency_index(matrix)	$O(n^2)$	A loop and a nested loop iterating over the matrix

functions like *xu_and_wei_cr, cao_cr,* and *szybowski_cr*, despite being central to the library's functionality, manifest a higher computational complexity of $O(n^4)$, rendering them less efficient as the size of the input matrix increases. A detailed perspective on the performance of these functions can be gleaned from the empirical data compiled in Table 1.

Performance testing of inconsistency reduction algorithms included in the library

An examination of the performance of inconsistency reduction (to a CR level of less than 0.1, which is considered sufficient) of 10,000 random pairwise comparison matrices ranging in size from 3×3 to 10×10 was performed. Timing calculations were performed using the *timeit* library in *Python*, using the *Google Colaboratory* platform, with the parameters of a single core *Intel Xeon*® CPU 2.20GHz, 12.7GB of RAM and 100GB of disk space. The calculations were repeated 5 times for each case and an average was drawn from them, which is included in Table 2 and visualized in the graph in Figure 2. The provided results show the execution times for three different algorithms included in the library and it can be observed that as the matrix size increases, the execution times for all three algorithms also increase. This is expected since larger matrices require more computational effort to process. Additionally, it can be concluded that the Python library reduce.py offers significantly faster execution times compared to spreadsheets. While spreadsheets might be quicker for small-scale operations involving one or two matrices, the creation of a script using the reduce.py library can provide substantial benefits when dealing with hundreds, thousands, or even tens of thousands of matrices. The advantage of using reduce.py lies in the ability to automate and

Table 2. Inconsistency reduction functions execution time comparison for different matrix sizes

Matrix size	xu_and_wei_cr (ms)	szybowski_cr (ms)	cao_cr (ms)
3×3	978	492	312
4×4	2.370	1.061	558
5×5	4.634	1.937	954
6×6	8.023	3.128	1.539
7×7	12.862	4.843	2.341
8×8	19.611	7.132	3.394
9×9	27.433	9.988	4.714
10×10	37.887	13.424	6.322

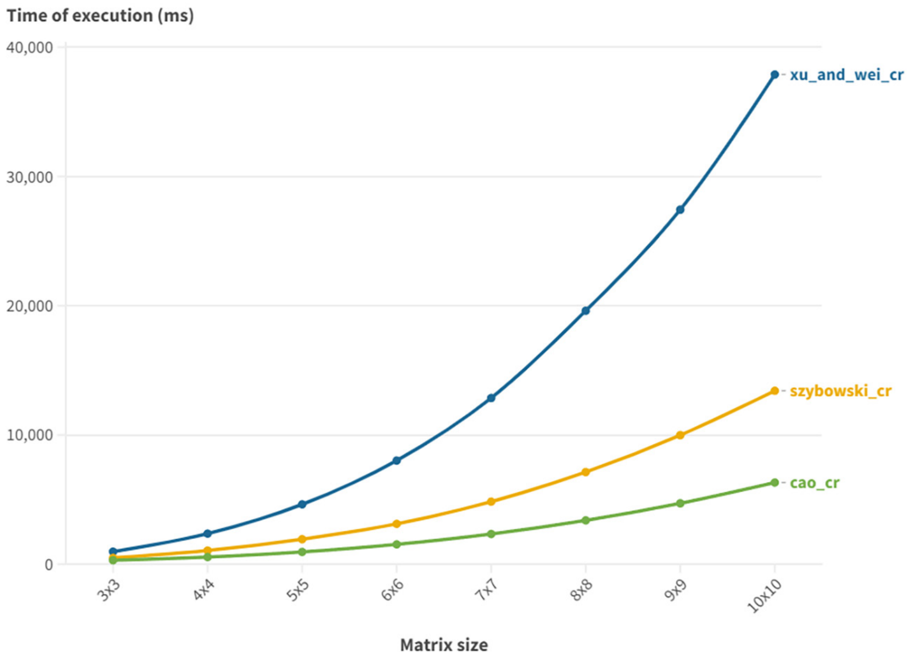

Fig. 2. Inconsistency reduction functions execution time comparison for different matrix sizes

efficiently process a large number of matrices in a consistent and time-efficient manner. The simplicity of the Python language also contributes to the ease of creating such scripts, allowing researchers or users to spend the same amount of time while achieving tangible results.

Illustrative example

In order to show the operation of this library in practice Listing 1 presents an example application in Python, which, implementing the library presented in this paper, will generate a random comparison matrix of size 6×6, and then reduce its consistency ratio (CR) to a value below 0.1, using all the algorithms available in the library.

IMPACT

As shown in the Section 3, the solution proposed in this paper allows implementing ready-made and researcher-tested algorithms in programs that allow greater freedom in matters of data loading and code customization. The implementation of the above algorithms in Python has resulted in a number of research papers and application tools using this – until now intended for in-ternal use – library. The Reduce library was first used during a Monte Carlo study of pairwise comparison matrix inconsistency reduction algorithms described by Mazurek et. al [20]. Along with this research work, a free CLI-type tool, PCM-CR [22], was published. The Reduce library is also a part of the back-end of a web application called REDUCE, which was described by Kowal et al [23] and which is the de facto graphical interface for this library accessible from any web browser. The REDUCE application is available at [24]. Also, based on the Reduce library, is the PC MATRICES GENERATOR tool described by Kuraś et al. [25], which uses pairwise com-parison algorithms to streamline the process of generating random comparison matrices with the desired consistency ratio (CR). The tool is available at [26].

Potential impact on the development of small and medium-sized enterprises, research and industrial applications

The appearance of the free pairwise comparison inconsistency reduction library with tools presented in this paper, and the fact that these applications are able to effectively count the most important matrix inconsistency indices, may enable small and medium-sized enterprises to take advantage of the capabilities of multi-criteria decision-making methods – until now they have had to use either cumbersome and time-consuming spreadsheets or paid software such as Expert Choice [43] to do so. Implementing decision-making methods is also an important part of the transition to Industry 4.0 [44], which, from the perspective of today's applications, has already had tangible effects in industrial applications as well [45, 46]. From a scientific point of view, the reduction of inconsistency in the elements of a pairwise comparison matrix is an important element of research in fuzzy logic [14], preference programming [15], or constructive consistent approximations [45].

CONCLUSIONS

In this study, a Python library for reducing pairwise comparison matrix inconsistencies – *reduce.py* – was created for the first time, available to the public and at no extra charge. This library can help improve research in topics such as AHP, BWM, or other decision support methods that use pairwise comparison matrices. Users can use the library either by importing it yourself into your Python project, or by using tools based on the library, presented in the Section 5. This publication demonstrates that this solution is more effective than other freely available tools for researchers and SMEs, in the form of spreadsheets such as LibreOffice or Excel, and with the ease of application and simplicity of the Python language, such entities will not be forced to use paid solutions such as Expert Choice to benefit from the capabilities of these methods. Although the library currently uses the most effective algorithms available at the time of publication (2023), it will be continuously developed by the authors with other new emerging algorithms, provided they have been sufficiently tested for effectiveness and efficiency. Of the non-iterative algorithms, it is worth mentioning a few of them: [32-42], which will become part of the REDUCE module in the next planned release as the project is still under development.

REFERENCES

1. Lull R. Artifitium electionis personarum (The method for the elections of persons) 1274-1283. Available from: https://www.math.uni-augsburg.de/htdocs/emeriti/pukelsheim/llull/
2. Condorcet Md. Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix, Paris, France; 1785.
3. Thurstone L.L. A law of comparative judgments. Psychological Reviews. 1927; 34: 273–286.
4. Saaty T.L. A Scaling Method for Priorities in Hierarchical Structures. Journal of Mathematical Psychology. 1977; 15: 234–281.
5. Vaidya O.S., Kumar S. Analytic hierarchy process: An overview of applications. European Journal of Operational Research. 2006; 169: 1–29.
6. Barzilai J. Consistency measures for pairwise comparison matrices. Journal of Multi-Criteria Decision Analysis. 1998; 7(3): 123–132.
7. Golden B., Wang Q. An alternate measure of consistency. In: Golden B, Wasil E, Harker PT, editors. The Analytic Hierarchy Process, Applications and Studies. Berlin–Heidelberg: Springer-Verlag; 1989; 68–81.
8. Koczkodaj W.W. A new definition of consistency of pairwise comparisons. Mathematical and Computer Modeling. 1993; 18(7): 79–84.
9. Obata T., Shiraishi S., Daigo M., Nakajima N. Assessment for an incomplete comparison matrix and improvement of an inconsistent comparison: computational experiments. ISAHP. Kobe, Japan; 1999.
10. Peláez J.I., Lamata M.T. A new measure of inconsistency for positive reciprocal matrices. Computer and Mathematics with Applications. 2003; 46(12): 1839–1845.
11. Aguarón J., Moreno-Jiménez J.M. The geometric consistency index: Approximated threshold. European Journal of Operational Research. 2003; 147(1): 137–145.
12. Aguarón J., Escobar M.T., Moreno-Jiménez J.M., Turón A. The Triads Geometric Consistency Index in AHP-Pairwise Comparison Matrices. Mathematics. 2020; 8: 926. https://doi.org/10.3390/math8060926.
13. Stein W.E., Mizzi P.J. The Harmonic Consistency Index for the Analytic Hierarchy Process. European Journal of Operational Research. 2007; 177(1): 488–497.
14. Ramík J., Korviny P. Inconsistency of pairwise comparison matrix with fuzzy elements based on geometric mean. Fuzzy Sets and Systems. 2010; 161: 1604–1613.
15. Salo A.A., Hämäläinen R. Preference Programming through Approximate Ratio Comparisons. European Journal of Operational Research. 1995; 82(3): 458–475.
16. Harris C.R., Millman K.J, van der Walt S.J., Gommers R., Virtanen P., Cournapeau D., et al. Array programming with NumPy. Nature. 2020; 585(7825): 357–362. https://doi.org/10.1038/s41586-020-2649-2
17. Virtanen P., Gommers R., Oliphant T.E., Haberland M., Reddy T., Cournapeau D., et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020; 17: 261–272. https://doi.org/10.1038/s41592-019-0686-2
18. Meurer A., Smith C.P., Paprocki M., Čertík O., Kirpichev S.B., Rocklin M., et al. SymPy: symbolic computing in Python. PeerJ Computer Science. 2017; 3: e103.
19. Department of Complex Systems, Rzeszow University of Technology. REDUCE [Internet]. GitHub. 2023. Available from: https://github.com/zszprz/reduce. DOI:10.5281/zenodo.7335819
20. Mazurek J., Perzina R., Strzałka D., Kowal B., Kuraś P. A numerical comparison of iterative algorithms for inconsistency reduction in pairwise comparisons. IEEE Access. 2021; 9: 62553-62561.
21. Mazurek J. Advances in Pairwise Comparisons: The Detection, Evaluation and Reduction of Inconsistency. Heidelberg: Springer; 2023.
22. Kuraś P. GitHub [Internet]. 2022. Available from: https://github.com/pawkuras/PCM_CR
23. Kowal B., Kuraś P., Strzałka D., Mazurek J., Perzina R. REDUCE. 3rd International conference on Decision making for Small and Medium-Sized Enterprises DEMSME 2021. Conference Proceedings.
24. Kowal B., Kuraś P., Strzałka D., Mazurek J., Perzina R. REDUCE [Internet]. 2021. Available from: https://reduce.prz.edu.pl/
25. Kuraś P., Gerka A. Using inconsistency reduction algorithms in comparison matrices to improve the performance of generating random comparison matrices with a given inconsistency coefficient range. Advances in Science and Technology Research Journal. 2023; 17(1): 222-229.
26. Kuraś P. Matrices generator [Internet]. 2022. Available from: https://reduce.prz.edu.pl/pc_matrices_generator
27. Cao D., Leung L.C., Law J.S. Modifying Inconsistent Comparison Matrix in Analytic Hierarchy Process: A Heuristic Approach. Decision Support Systems. 2008; 44: 944–953. DOI: 10.1016/j.dss.2007.11.002
28. Szybowski J. The improvement of data in pairwise comparison matrices. Procedia Computer Science. 2018; 126: 1006–1013.
29. Zeshui X., Cuiping W. A consistency improving method in the analytic hierarchy process. European Journal of Operational Research. 1999; 116(2): 443–449.

30. Kou G., Ergu D., Shang J. Enhancing data consistency in decision matrix: Adapting Hadamard model to mitigate judgment contradiction. European Journal of Operational Research. 2014; 236(1): 261–271.
31. Mazurek J., Perzina R., Strzałka D., Kowal B. A new step-by-step (SBS) algorithm for inconsistency reduction in pairwise comparisons. IEEE Access. 2020; 8: 135821–135828.
32. Abel E., Mikhailov L., Keane J. Inconsistency reduction in decision making via multi-objective optimisation. European Journal of Operational Research. 2018; 267(1): 212–226.
33. Bozóki S., Fülöp J., Poesz A. On pairwise comparison matrices that can be made consistent by the modification of a few elements. Central European Journal of Operations Research. 2011; 19(2): 157–175.
34. Bozóki S., Fülöp J., Poesz A. On reducing inconsistency of pairwise comparison matrices below an acceptance threshold. Central European Journal of Operations Research. 2015; 23(4): 849–866.
35. Negahban A. Optimizing consistency improvement of positive reciprocal matrices with implications for Monte Carlo analytic hierarchy process. Computers & Industrial Engineering. 2018; 124: 113–124.
36. Gao J., Shan R. A new method for modification consistency of the judgment matrix based on genetic ant algorithm. Applied Mathematics & Information Sciences. 2012; 6(1): 35–39.
37. Girsang A.S., Tsai C.W., Yang C.S. Ant algorithm for modifying an inconsistent pairwise weighting matrix in an analytic hierarchy process. Neural Computing and Applications. 2015; 26(2): 313–327.
38. Zhang H., Sekhari A., Ouzrout Y., Bouras A. Optimal inconsistency repairing of pairwise comparison matrices using integrated linear programming and eigenvector methods. Mathematical Problems in Engineering. 2014.
39. Li H.L., Ma L.C. Detecting and adjusting ordinal and cardinal inconsistencies through a graphical and optimal approach in AHP models. Computers & Operations Research. 2007; 34(3): 780–798.
40. Ergu D., Kou G., Peng Y., Shi Y. A simple method to improve the consistency ratio of the pair-wise comparison matrix in ANP. European Journal of Operational Research. 2011; 213(1): 246–259.
41. Kułakowski K., Juszczyk R., Ernst S. A concurrent inconsistency reduction algorithm for the pairwise comparisons method. In International Conference on Artificial Intelligence and Soft Computing. Cham: Springer; 2015; 214–222.
42. Pereira V., Costa H.G. Nonlinear programming applied to the reduction of inconsistency in the AHP method. Annals of Operations Research. 2015; 229(1): 635–655.
43. Erdogan S.A., Šaparauskas J., Turskis Z. Decision making in Construction Management: AHP and Expert Choice approach. Procedia Engineering. 2017; 172: 270–276.
44. Sajjad A., Ahmad W., Hussain S. Decision-Making Process Development for Industry 4.0 Transformation. Advances in Science and Technology Research Journal. 2022; 16(3): 1–11. https://doi.org/10.12913/22998624/147237
45. Kozera R., Smarzewski R. Constructive Consistent Approximations in Pairwise Comparisons. Advances in Science and Technology Research Journal. 2022; 16(4): 243–255. https://doi.org/10.12913/22998624/153086
46. Stevanović D., Lekić M., Krzanovic D., Ristović I. Application of MCDA in Selection of Different Mining Methods and Solutions. Advances in Science and Technology Research Journal. 2018; 12(1): 171-180. https://doi.org/10.12913/22998624/85804
47. Urbański T. Assessment of the productibility of hybrid nodes using the multi-criteria method. Advances in Science and Technology Research Journal. 2014; 8(22): 31–36. https://doi.org/10.12913/22998624.1105160