Novel Fuzzy MCDM Model for Comparison of Programming Languages

Srđan DAMJANOVIĆ[1], Predrag KATANIĆ[1], Edmundas Kazimieras ZAVADSKAS[2]*, Željko STEVIĆ[3,4], Branko KRSMANOVIĆ[1], Nataša DJALIĆ[3]

[1] Faculty of Business Economics, University of East Sarajevo, Bijeljina, 1C Semberskih Ratara, 76300, Bijeljina, Bosnia and Herzegovina
srdjan.damjanovic@fpe.ues.rs.ba, predrag.katanic@fpe.ues.rs.ba, branko.krsmanovic@fpe.ues.rs.ba

[2] Institute of Sustainable Construction, Vilnius Gediminas Technical University, LT-10223 Vilnius, Lithuania
edmundas.zavadskas@vilniustech.lt (**Corresponding author*)

[3] Faculty of Transport and Traffic Engineering, University of East Sarajevo, 52 Vojvode Mišića, 74000 Doboj, Bosnia and Herzegovina
zeljko.stevic@sf.ues.rs.ba, natasa.djalic@sf.ues.rs.ba

[4] College of Engineering, Korea University, 145 Anam-Ro, Seongbuk-Gu, 02841 Seoul, Republic of Korea
172317@korea.ac.kr

Abstract: In the last twenty years, a large number of new programming languages have emerged, along with the modernization and evolution of existing ones. The selection of an appropriate programming language to solve a specific problem is often a topic of debate among university professors and programmers. The development of new programming languages is also influenced by advancements in computer hardware, communication, and measurement equipment. This paper aims to explore methods for qualitative comparison of different programming languages. A comparative analysis is presented among four programming languages: VisualBasic, C++, Python, and VEEPro. A new extension of OPARA (Objective Pairwise Adjusted Ratio Analysis) was created for ranking programming languages, while the IMF SWARA (Improved Fuzzy Stepwise Weight Assessment Ratio Analysis) was used to determine the importance of 10 criteria. Since this involves group decision-making with the participation of 20 experts, their preferences were averaged using the Bonferroni operator. The results of the created model show that C++ is the programming language with the best performance among the set of considered alternatives and criteria. To validate the model, verification analyses were defined, confirming the initial results.

Keywords: Fuzzy OPARA, Programming Language, IMF SWARA, MCDM.

1. Introduction

Programming languages are an inexhaustible topic of debate in professional circles today, discussing about which programming language is better and why a particular programmer or company has chosen to write a program in it. Scientists from various fields often use different programming languages, claiming that the one they use is the best. Programmers frequently create new programming languages in order to solve practical problems more easily or quickly. Thousands of different programming languages have been created in recent years. Some languages are more commonly used in the education of programmers, while others are utilized in specific areas of science and research. Each language has its own advantages and disadvantages. This paper provides a comparative analysis of various properties and characteristics of some popular programming languages: VisualBasic, C++ and Python, as well as the VEEPro programming language, which is highly specialized for communication with programmable measurement instruments. VEEPro is a programming language primarily designed for programming connection between computers and measurement instruments from Hewlett Packard, then Agilent, and now Keysight. However, this programming language is also successfully used for programming connection with programmable measurement instruments from other well-known global manufacturers. Programming in the VEEPro programming language is done using ready-made objects that are placed on the workspace and then connected to each other by drawing lines dragging the mouse from the output of one object to the input of another. This diversity of programming languages primarily aims to show teachers who teach programming in schools and universities the fundamental differences in syntax among programming languages and which programming languages are simpler in the initial stages of learning programming. For the purposes of this paper, programs are written to compare the execution speed of these programs for a large number of cycles. Programs for adding and dividing numbers, calculating the sine of an angle, computing the logarithm of natural numbers, sorting an array of numbers, as well as writing and reading data into a text file and an Excel document are created. For each program, a series of 10 measurements of program duration was conducted and the standard deviation

https://doi.org/10.24846/v33i4y202401

of the dispersion of time measurement results, as well as the relative deviation of the standard deviation from the mean value of measurements were calculated. Multi-criteria decision-making (MCDM) represents a field that is crucial for making precise and correct decisions in various fields of activity, as supported by studies (Petrovas et al., 2023; 2024; Baydas et al., 2024).

After the introduction, the paper is structured into five other sections. Section 2 presents a review of the current state in the field from the point of view of programming language analysis. Section 3 provides a detailed development and extension of the OPARA method with TFNs, while Section 4 introduces the formation of the MCDM model, a team of experts for group decision-making, as well as the final results of the model. In Section 5, a comparative analysis and an examination of the impact of simulated values of weighting coefficients are conducted. Finally, the last section Section 6, summarizes the findings through the conclusion.

2. Literature Review

Since the advent of computers and programming languages, many programmers have been engaged in comparing programming languages. This paper will provide a brief overview of some articles that have addressed the comparison of various programming languages. The study by Naim et al. (2010) presents a survey focusing on a selection of programming languages, including C++, C#, Java, JavaScript, AspectJ, Haskell, PHP, Scala, Scheme, and BPEL. The study provides a comparative analysis of these ten languages, evaluated against several criteria such as secure programming practices, web application development, web service composition, object-oriented programming (OOP) abstractions, reflection, aspect-oriented programming, functional programming, declarative programming, batch scripting, and user interface prototyping. The languages are analyzed based on these parameters and the degree of support each provides to the others.

The article of Stein & Geyer-Schulz (2013) examined C++, Java, C#, F#, and Python in a controlled environment where a graph clustering task was developed and executed for each language. Their paper outlines the problem being addressed and presents an in-depth exploration of the various characteristics of the selected languages. The results reveal that C++ offers the highest performance for the given task, though Java, C#, and F# perform similarly under specific conditions. Additionally, the study highlights the potential for modern languages like Python and F# to reduce the overall codebase size.

In the paper of Zakaria et al. (2015), a comparative analysis of six programming languages, namely C++, PHP, C#, Java, Python, and Visual Basic (VB) is conducted. The study evaluates these languages based on various attributes, including reusability, portability, reliability, availability of compilers and tools, familiarity, efficiency, readability, and expressiveness.

The research conducted by Parveen & Fatima (2016) presents a comparative analysis of three widely utilized programming languages: Java, C#, and C++. The comparison is made with respect to several criteria, including syntax, number lines of code, compilation time, machine dependency, execution time, speed (or efficiency), and flexibility. The authors investigate these languages in relation to the specified criteria, assessing the degree of support each language provides to the others. The paper's objective is to deliver a thorough comparative understanding of the selected languages. By employing a straightforward common program across all languages, the researchers facilitate effective comparisons. Ultimately, the goal is to assist programmers in recognizing and exploring the most favorable attributes of high-level programming languages.

In his paper, Pejovic (2019) explores the use of the Python programming language for automating measurement systems and developing virtual measurement instruments. The study highlights the availability of various Python modules for specific tasks, which can be sourced from the Python Standard Library or external repositories, with some modules specifically designed for facilitating communication with instruments. The applications of Python began in the field of power electronics and have successfully expanded into metrology, the design of a DC voltage calibrator, and educational contexts. The methodologies were utilized to modernize two courses and for measuring system frequency response in electronics, acoustics, and control system design. The study concludes that Python serves as an effective tool for building automated measurement systems and virtual instruments, thanks to its modular architecture and the flexibility it offers for module contributions. When choosing a programming tools, the focus has been on

promoting general-purpose tools and techniques to reduce the need for specialized knowledge.

The process of selecting a programming language is fundamentally an MCDM (Multi-Criteria Decision-Making) problem, as noted by Mishra et al. (2020). To address the uncertainties inherent in MCDM scenarios, fuzzy set theory serves as a valuable tool. Interval-valued intuitionistic fuzzy sets (IVIFSs) offer enhanced flexibility in modeling uncertainty compared to traditional fuzzy sets (FSs), as they incorporate interval membership, non-membership, and hesitancy functions. This paper introduces a novel integrated approach based on the multi-attributive border approximation area comparison (MABAC) method for tackling MCDM problems utilizing IVIFSs. The proposed method incorporates IVIFS operators, modifies the classical MABAC approach, and introduces a new process for calculating criteria weights. To determine these weights, the subjective assessments from decision experts are combined with objective weights derived from a proposed entropy measure and divergence techniques, yielding more realistic weight values. The analysis demonstrates that merging subjective and objective weights can enhance the stability of the proposed method across varying criteria weights. Additionally, the paper compares the outcomes of the proposed approach with existing methods to validate its effectiveness. This comparative analysis reveals that the proposed method is both efficient and aligns well with other methodologies.

In their research, Pereira et al. (2021) conduct a comparative analysis of a wide range of programming languages with a focus on their efficiency from an energy consumption perspective. Their objective is to develop and evaluate various rankings for programming languages based on energy efficiency metrics. Their paper applied rigorous and systematic methodologies to address ten well-defined programming challenges, utilizing 27 programming languages sourced from the renowned Computer Language Benchmark Game repository. The findings of their study yield significant insights, including how the energy consumption patterns of slower and faster languages may differ, as well as the impact of memory usage on overall energy consumption. Additionally, they offer a straightforward approach for leveraging these results to assist software engineers and practitioners in making informed choices regarding programming languages when energy efficiency is a critical consideration.

In his paper, Filip (2021) provides a concise historical overview of the role that computers and automation have played in fostering new working methods while simultaneously enhancing the quality of life for individuals. He emphasizes that this evolution has been further accelerated by the current efforts people are undertaking to adapt to and alleviate the negative effects of the ongoing pandemic, striving to return to a state of normalcy.

In their research, Ali & Qayyum (2021) concentrate on the four programming languages: C, C++, Python, and Java, evaluating them based on the criteria of time, speed, and simplicity. They utilize a single optimized pseudo-code example to implement code in each of these programming languages, adhering to their specific syntax and conventions. The findings of this comparison are summarized in a table to facilitate the understanding of the results for the reader.

Chiu et al. (2022) conduct a thorough performance analysis and comparison of Java, JavaScript, Go, and Python, utilizing C++ as a reference point. They meticulously instrument the runtimes of these languages, allowing developers to accurately gauge the cycle count required to execute any bytecode instruction or to determine the overhead associated with dynamic type checking in JavaScript. This methodology facilitates the precise identification of overhead sources. The authors provide a comprehensive examination of completion times, resource utilization, and scalability across the analyzed languages.

Languages developed in the 1950s, such as Fortran, remain in active use today, owing to their versatility and foundational role in supporting a substantial portion of the legacy systems and applications in our digital landscape. Since that era, numerous additional languages have emerged, exhibiting increasing diversity over time. Modern languages such as C, C++, Python, Java, JavaScript, and PHP have greatly enhanced efficiency and are utilized across a wide array of applications. Sakharkar (2023) emphasizes that the proliferation of programming languages not only enhances accessibility but also exposes applications to significant security challenges. These security concerns differ in terms of prevalence and language specificity.

Zakeri-Nasrabadia et al. (2023) conduct a systematic literature review and meta-analysis focused on techniques for measuring and evaluating code similarity, aiming to clarify

existing methodologies and their attributes across various applications. Their research began with a search of four digital libraries, resulting in an initial pool of over 10,000 articles, which was narrowed down to 136 primary studies relevant to the topic. A thorough examination of these studies reveals 80 software tools that employ eight distinct techniques across five application domains. Notably, nearly 49% of these tools are tailored for Java programs, while 37% are compatible with C and C++.

In contrast, numerous programming languages receive insufficient support in this area. Jäger & Gümmer (2023) present PythonDAQ, an open-source Python package designed for the acquisition, visualization, storage, and post-processing of measurement data. This code enables the acquisition of data from any sensor with digital output, facilitates online calculations, and allows for the storage of both measured and computed data. In the concluding section, the authors compare PythonDAQ with commercial data acquisition (DAQ) solutions and the data acquisition software utilized by a leading aero-engine manufacturer.

In their study, Alves et al. (2023) analyze the typical linguistic features found in six prominent programming languages: C, C++, C#, Java, Python, and Haskell. Then they design and implement a survey aimed at gaining a deeper understanding of the intrinsic relationship between programmers and their primary tools programming languages. To begin interpreting the results, the authors focus on the first question, which assesses respondents' familiarity with the selected languages. Java, C, and Python emerged as the most recognized, with 95%, 88%, and 93% of participants indicating they could use these languages to some extent. This suggests that inquiries related to Java, C, and Python may yield more significant insights into their defining characteristics. In contrast, familiarity with Haskell, C++, and C# was notably lower, with only 47%, 45%, and 31% of respondents, respectively, claiming they were at least somewhat proficient in these languages.

Katzy et al. (2023) report that experiments indicate a close proximity in token representation among programming languages such as C++, Python, and Java, while tokens in languages like Mathematica and R demonstrate considerable dissimilarity. Their findings imply that such variations may lead to performance issues when handling multiple languages. Consequently, the authors advocate for the utilization of their similarity measure to guide the selection of a diverse range of programming languages during the training and evaluation of future models.

Padilla et al. (2023) explore the application of ChatGPT in the educational sector, noting its potential to significantly enhance teaching and learning processes. However, they identify a gap in understanding its implementation within programming courses in computing programs. This study examines the advantages, challenges, and issues associated with utilizing ChatGPT, a language AI model, for programming tasks. The findings highlight several benefits, such as improved coding efficiency, assistance in comprehending complex code, and its role as a problem-solving tool. Conversely, the research underscores significant challenges, including concerns about data privacy, ethical implications, tendencies toward plagiarism, and limitations in contextual comprehension.

In their study, Sambucci et al. (2023) analyze research showing that while AI enhances the efficiency of critical infrastructures, it also introduces vulnerabilities to sophisticated attacks. To address these risks, the authors emphasize the need for proactive security measures, including incident response and collaboration, to safeguard AI-driven systems.

3. Extension of the OPARA Method with TFNs

The OPARA method is a new MCDM tool developed by Keshavarz-Ghorabaee et al. (2024), which has been modified and extended with TFNs (triangular fuzzy numbers) in this paper. The algorithm of the Fuzzy OPARA method is presented through the following steps.

Step 1: Creation of the initial fuzzy decision matrix with elements a_{ij}

Step 2: Calculation of parameters based on which the subsequent normalization of the initial fuzzy decision matrix can be performed, whereby the values $\alpha = 5$ and $\beta = 0.7$.

a. calculation of values that represent the difference between the maximum and minimum values $\left(\overline{m_j}\right)$, the sum of the minimum and maximum values, $\left(\overline{m_j}'\right)$, and the quotient of these two previously calculated values $\left(\overline{b_j}\right)$.

$$\overline{m_j} = \left(m_j^{\,l}, m_j^{\,m}, m_j^{\,u}\right) = \max_i \overline{a_{ij}} - \min_i \overline{a_{ij}} \quad (1)$$

$$\overline{m_j}' = \left(m'_j{}^{\,l}, m'_j{}^{\,m}, m'_j{}^{\,u}\right) = \max_i \overline{a_{ij}} + \min_i \overline{a_{ij}} \quad (2)$$

$$\overline{b_j} = \frac{\overline{m_j}}{\overline{m_j}'} = \left(\frac{m_j^{\,l}}{m'_j{}^{\,u}}, \frac{m_j^{\,m}}{m'_j{}^{\,m}}, \frac{m_j^{\,u}}{m'_j{}^{\,l}}\right) \quad (3)$$

b. calculation of the value $\overline{c_j}$ that represents the product $\left(\overline{m_j}'\right)$ and $(\alpha - 1)$, $\overline{d_j}$ which represents the product $\alpha \otimes \max_i \overline{a_{ij}}$ and $\overline{e_j}$ as a quotient of two previously calculated values.

$$\overline{c_j} = (\alpha - 1) \otimes \max_i \overline{a_{ij}} + \min_i \overline{a_{ij}} \quad (4)$$

$$\overline{d_j} = \alpha \otimes \max_i \overline{a_{ij}} \quad (5)$$

$$\overline{e_j} = \frac{\overline{c_j}}{\overline{d_j}} = \left(\frac{c_j^{\,l}}{d_j^{\,u}}, \frac{c_j^{\,m}}{d_j^{\,m}}, \frac{c_j^{\,u}}{d_j^{\,l}}\right) \quad (6)$$

c. calculation of the value $\overline{f_j}$ which is the adjustment parameter determined using an adjustment function based on the corresponding performance range for each criterion.

$$\overline{f_j} = \begin{cases} \overline{e_j} & if\ \overline{b_j} > \beta \\ 1 & otherwise \end{cases} \quad (7)$$

Step 3: In this step, it is necessary to modify the original OPARA method by introducing the normalization process. The normalized values are shown as $\overline{g_{ij}}$ for both types of criteria: cost (C) and benefit (B).

$$\overline{g_{ij}} = \overline{w_j}\left(\frac{\overline{a_{ij}}}{\max_i \overline{a_{ij}}}\right)^{\overline{f_j}} = \begin{pmatrix} w_j^{\,l}\left(\frac{a_{ij}^{\,l}}{\max a_{ij}^{\,u}}\right)^{f_j^{\,l}}, \\ w_j^{\,m}\left(\frac{a_{ij}^{\,m}}{\max a_{ij}^{\,m}}\right)^{f_j^{\,m}}, \\ w_j^{\,u}\left(\frac{a_{ij}^{\,u}}{\max a_{ij}^{\,l}}\right)^{f_j^{\,u}} \end{pmatrix} if\ j \in B \quad (8)$$

$$\overline{g_{ij}} = \overline{w_j}\left(\frac{\max_i \overline{a_{ij}}}{\overline{a_{ij}}}\right)^{\overline{f_j}} = \begin{pmatrix} w_j^{\,l}\left(\frac{\max a_{ij}^{\,l}}{a_{ij}^{\,u}}\right)^{f_j^{\,l}}, \\ w_j^{\,m}\left(\frac{\max a_{ij}^{\,m}}{a_{ij}^{\,m}}\right)^{f_j^{\,m}}, \\ w_j^{\,u}\left(\frac{\max a_{ij}^{\,u}}{a_{ij}^{\,l}}\right)^{f_j^{\,u}} \end{pmatrix} if\ j \in C \quad (9)$$

Step 4: In this step, a parameter $\overline{h_j}$ is introduced, defined as (1,1,1) by decision-makers, based on the original OPARA method from the paper (Keshavarz-Ghorabaee et al., 2024). After that, the following equations are defined, again depending on the type of criteria.

$$\overline{i_{ij}} = \overline{w_j}\left(\frac{\overline{a_{ij}}}{\max_i \overline{a_{ij}}}\right)^{\overline{h_j}} = \begin{pmatrix} w_j^{\,l}\left(\frac{a_{ij}^{\,l}}{\max a_{ij}^{\,u}}\right)^{h_j^{\,l}}, \\ w_j^{\,m}\left(\frac{a_{ij}^{\,m}}{\max a_{ij}^{\,m}}\right)^{h_j^{\,m}}, \\ w_j^{\,u}\left(\frac{a_{ij}^{\,u}}{\max a_{ij}^{\,l}}\right)^{h_j^{\,u}} \end{pmatrix} if\ j \in B \quad (10)$$

$$\overline{i_{ij}} = \overline{w_j}\left(\frac{\max_i \overline{a_{ij}}}{\overline{a_{ij}}}\right)^{\overline{h_j}} = \begin{pmatrix} w_j^{\,l}\left(\frac{\max a_{ij}^{\,l}}{a_{ij}^{\,u}}\right)^{h_j^{\,l}}, \\ w_j^{\,m}\left(\frac{\max a_{ij}^{\,m}}{a_{ij}^{\,m}}\right)^{h_j^{\,m}}, \\ w_j^{\,u}\left(\frac{\max a_{ij}^{\,u}}{a_{ij}^{\,l}}\right)^{h_j^{\,u}} \end{pmatrix} if\ j \in C \quad (11)$$

Step 5: Creation of the aggregated value $\overline{k_j}$ between $\overline{g_{ij}}$ and $\overline{i_{ij}}$ by introducing a coefficient ω which ranges from 0-1.

$$\overline{k_{ij}} = \left(\omega \otimes \overline{g_{ij}}\right) + \left((1-\omega) \otimes \overline{i_{ij}}\right) \quad (12)$$

Step 6: Summing the elements of the fuzzy matrix $\overline{k_{ij}}$ by columns, obtaining then:

$$\overline{l_j} = \sum_{j=1}^{m} \overline{k_{ij}} \quad (13)$$

where m is the number of criteria.

Step 7: Ranking the alternative solutions based on the final values obtained as follows:

$$\overline{o_i} = \frac{1}{n} \otimes \sum\left(\frac{\overline{k_{ij}}}{\overline{l_j}}\right) = \frac{1}{n} \otimes \sum\left(\frac{k_{ij}^{\,l}}{l_j^{\,u}}, \frac{k_{ij}^{\,m}}{l_j^{\,m}}, \frac{k_{ij}^{\,u}}{l_j^{\,l}}\right) \quad (14)$$

where n is the number of alternatives, and the highest value represents the best-ranked alternative.

4. Forming a MCDM Model

Due to the inability to present all experimental measurements and results for the eighth, ninth and tenth criteria, a brief description of the measurement procedure is provided below. The

mentioned measurements refer to the results of comparing the execution speed for the same program code in four programming languages, which were compared with each other. Each program was run ten times on the same computer using the Windows operating system. Based on the ten measurements of program execution time, the mean value and standard deviation for the ten measurements were calculated. The measurements displayed the comparative results of measuring the time taken to perform addition, division of two numbers, calculate the logarithm base 10, compute the sine of an angle in a double for loop, and write, read and sort a series of numbers.

After defining all the measurements and providing the comparative values for the four programming languages, which essentially represent alternatives in this MCDM model: VisualBasic (A1), C++ (A2), Python (A3), and VEEPro (A4), a list of criteria was defined on the basis of which the programming languages were evaluated, as shown in Table 1.

In the study comparing four programming languages, a total of 20 experts participated, including 9 university professors, 6 university associates and 5 experts from industry who are engineers.

Among experts, 3 hold the title of full professor, 5 are associate professors, 1 is an assistant professor, 4 are senior assistants, and 2 are assistants. Some of the university experts are engaged in pure programming, while others work in electrical measurements and use various programming languages to manage measuring instruments and devices. The experts from industry are engineers who deal with programming related to measuring devices and control systems. The range of work experience of engineers is from 10 to 30 years, and they have used various programming languages in their work so far. Each expert conducted an individual evaluation of the importance of the criteria, and later the programming languages as potential alternatives. To determine the weights of the criteria, the IMF SWARA was applied (Vrtagić et al., 2021; Stević et al., 2022; Badi & Bouraima, 2023) in combination with the fuzzy Bonferroni operator (Hadžikadunic et al., 2023; Radovanović et al., 2023), resulting in the following calculated weights:

w_1=(0.093,0.102,0.112),w_2=(0.103,0.112,0.122),
w_3=(0.123,0.132,0.142),w_4=(0.087,0.097,0.107),
w_5=(0.098,0.107,0.118),w_6=(0.084,0.095,0.106),
w_7=(0.081,0.091,0.102),w_8=(0.093,0.103,0.113),
w_9=(0.068,0.078,0.089),w_{10}=(0.056,0.066,0.076)

This means that the third criterion, which represents development speed, is the most dominant in the eyes of programmers, while the tenth criterion, which refers to the speed of working with large datasets, is the least important. The next phase of applying the MCDM model involves processing data for the evaluation of programming languages by 20 experts who participated in the group decision-making. The

Table 1. Title and brief description of 10 criteria for comparing programming languages

	Name and Brief Description of Criteria
C1	Popularity - which of the considered programming languages are studied at the surveyed faculties.
C2	Ease of Learning - how quickly or easily things that have never been used before can be understood.
C3	Development Speed - how quickly a programmer can create a program. This usually shows how fast a programmer can code using a specific programming language.
C4	Readability – it refers to how easily a reader comprehends a written text. In the context of natural language, the readability of a text is influenced by its content, the complexity of vocabulary and syntax, as well as by its presentation, which includes typographical elements such as font size, line height, and line length.
C5	Efficiency - the capacity to minimize the waste of materials, energy, time, and money during the development of a program. More generally, it refers to the ability to do the job well, successfully, and without wasting time.
C6	Tool Support - the extent to which a programming language provides ready-made programming tools.)
C7	Portability - a parameter of a computer program that indicates whether the program can be used on multiple operating systems. Portability is essentially the task of adapting any job required to build a computer program that operates in a new environment.
C8	Speed of Mathematical Function Processing
C9	Speed of Data Writing and Reading
C10	Speed of Working with Large Datasets

evaluation was conducted using a linguistic scale in order to create a conversion to TFNs through processing. The obtained values of the initial TFN matrix (Table 2) were also averaged using the fuzzy Bonferroni operator.

In the following, the algorithm of the fuzzy OPARA method is explained through a partial calculation. First, the values are calculated:

$$\overline{m_1} = (2.973, 4.373, 5.787) =$$
$$(5.583 - 2.610, 6.586 - 2.213, 7.558 - 1.801)$$

$$\overline{m_1}' = (7.385, 8.799, 10.198) =$$
$$(5.583 + 1.801, 6.586 + 2.213, 7.588 + 2.610)$$

$$\overline{b_1} = (0.292, 0.497, 0.784) = \left(\frac{2.973}{10.198}, \frac{4.373}{8.799}, \frac{5.787}{7.385}\right)$$

$$\overline{c_1} = (24.135, 28.558, 32.962)$$

$$\overline{d_1} = (27.917, 32.930, 37.940)$$

$$\overline{e_1} = (0.636, 0.867, 1.181) = \left(\frac{24.135}{37.940}, \frac{28.558}{32.930}, \frac{32.962}{27.917}\right)$$

$\overline{f_1} = (1.000, 1.000, 1.181)$ since

$\overline{b_1} = (0.292, 0.497, 0.784)$, which means that l and m of this number are $< \beta = 0.7$, and $u > \beta = 0.7$. It is important to note that all criteria are of the benefit type, so Equation (8) is applied for normalization.

$$\overline{g_{11}} = \left(0.093\left(\frac{2.254}{7.588}\right)^1, 0.102\left(\frac{2.871}{6.586}\right)^1, 0.112\left(\frac{3.474}{5.583}\right)^{1.181}\right)$$

$$\overline{i_{11}} = \left(0.093\left(\frac{2.254}{7.588}\right)^1, 0.102\left(\frac{2.871}{6.586}\right)^1, 0.112\left(\frac{3.474}{5.583}\right)^1\right)$$

$$\overline{k_{11}} = (0.027, 0.044, 0.067) =$$
$$(0.5 \otimes 0.027) + ((1 - 0.5) \otimes 0.027)$$
$$(0.5 \otimes 0.044) + ((1 - 0.5) \otimes 0.044)$$
$$(0.5 \otimes 0.064) + ((1 - 0.5) \otimes 0.070)$$

$$\overline{l_1} = (0.179.0.274, 0.418) =$$
$$(0.027 + 0.068 + 0.062 + 0.022)$$
$$(0.044 + 0.102 + 0.094 + 0.034)$$
$$(0.067 + 0.157 + 0.145 + 0.049)$$

$$\overline{o_1} = (0.224, 0.561, 1.387)$$

After the calculation was complete, the obtained results are shown in Table 3.

The obtained results indicate that within the defined set of elements of the MCDM model, C++ represents the programming language with the best-rated performance by the experts involved in the research.

5. Sensitivity and Comparative Analysis

Through this section of the study, an analysis of the change in the impact of weighting coefficients (Puška et al., 2024) on the ranking of programming languages was created, as such analysis provides decision-makers with the opportunity to make a better decision. This allows for the consideration of certain future impacts and the modeling of results accordingly.

Table 2. Initial TFN matrix in the Fuzzy OPARA method

	C1			C2			…	C9			C10		
A1	2.254	2.871	3.474	3.947	4.958	5.965		4.940	5.942	6.943	2.531	3.549	4.559
A2	5.583	6.586	7.588	4.878	5.827	6.775		5.095	6.096	7.096	5.046	6.047	7.047
A3	5.063	6.069	7.074	5.173	6.178	7.181	. . .	5.301	6.315	7.325	3.284	4.288	5.290
A4	1.801	2.213	2.610	3.333	4.172	5.005		2.830	3.717	4.603	4.707	5.652	6.596
max	5.583	6.586	7.588	5.173	6.178	7.181		5.301	6.315	7.325	5.046	6.047	7.047
min	1.801	2.213	2.610	3.333	4.172	5.005		2.830	3.717	4.603	2.531	3.549	4.559

Table 3. Results of the Fuzzy OPARA method

				Crisp	Rank
A1	0.224	0.561	1.387	0.724	3
A2	0.334	0.780	1.868	0.994	1
A3	0.305	0.720	1.730	0.918	2
A4	0.173	0.439	1.087	0.566	4

Figure 1 shows the weights of the parameters across 100 scenarios in which different values of the criteria are simulated in such a way that the weighting coefficients change from 5% to 95%.

It is important to note that there are no changes in the sensitivity analysis, i.e., that the initial results are identical across all 100 scenarios. This result is a consequence of the small number of alternative solutions on the one hand, and of the double-digit number of criteria on the other.

The next step involves creating a comparative analysis (Figure 2) with four other MCDM methods: Fuzzy Range of Value ROV (Ristić et al., 2024), Fuzzy MARCOS Measurement Alternatives and Ranking according to Compromise Solution (Damjanović et al., 2022), Fuzzy Simple Additive Method (Kabassi et al., 2020), and Fuzzy WASPAS weighted aggregated sum product assessment (Turskis et al., 2015).

The results of the comparison with additional four MCDM methods show the stability of the results and an identical ranking to that of the Fuzzy OPARA method.

In addition, it is important to emphasize that the analysis of the change in the coefficient ω was also conducted across all integer values in the range within 0-1, and that the results remained unchanged.

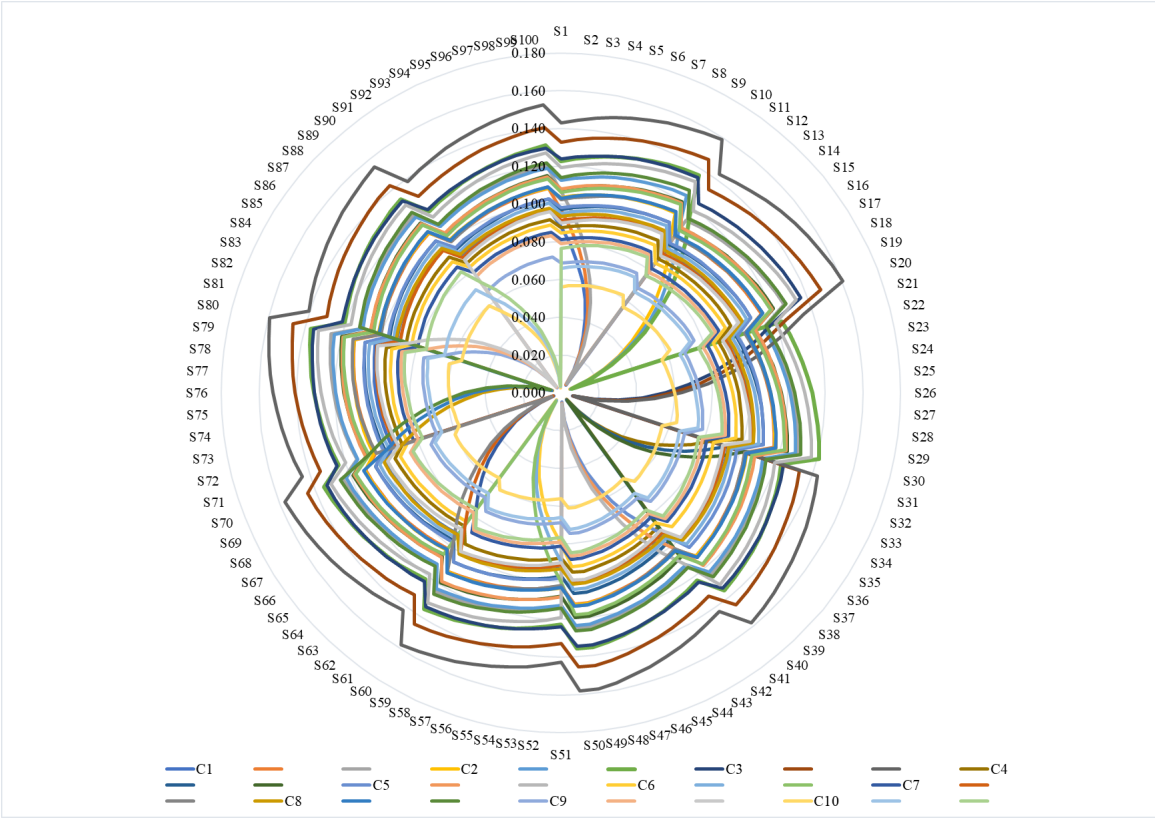

Figure 1. Values of weighting coefficients through 100 simulated scenarios

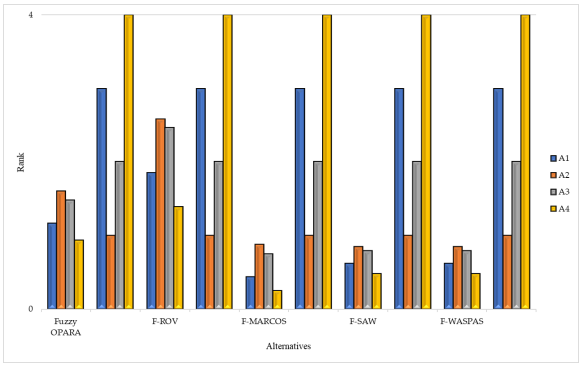

Figure 2. Ranks in comparative analysis

6. Conclusion

Through the research presented in this paper, four programming languages were analyzed by comparing the performance measurements of various operations. Based on this, certain quantitative indicators were obtained and included in a list of 10 criteria used by 20 experts to evaluate the programming languages. The contribution of this paper is reflected in the modification and extension of the OPARA method with TFNs, as well as in the extensive comparative analysis of programming languages, which partly serves as input parameters in the MCDM model, as decision-makers were provided with the results of the performance measurements of the operations. The expert team and the applied Fuzzy OPARA method demonstrate that, among the considered set of languages, C++ shows the best performance according to the established criteria, which was verified through sensitivity analysis, comparative analysis with four other MCDM methods, and variations in the coefficient ω. Future research based on this study implies the expansion of the list of experts in group decision-making, a greater number of simulation measurements related to the performance of specific operations, as well as the possibility of applying other evaluation tools.

REFERENCES

Ali, S. & Qayyum, S. (2021) *A Pragmatic Comparison of Four Different Programming Languages*. University of Management and Technology, Daska Campus, pp. 1-15. doi: 10.14293/S2199-1006.1.SOR-.PP5RV1O.v1.

Alves, J., Neto, A. C., Pereira, M. J. V. & Henriques, P. R. (2023) Characterization and Identification of Programming Languages. In: *Proceedings of 12th Symposium on Languages, Applications and Technologies (SLATE 2023), 26-28 June 2023, Vila do Conde, Portugal*. Dagstuhl Publishing, Saarbrücken/ Wadern, Germany, OASICS. pp. 1-13. doi: 10.4230/ OASIcs.SLATE.2023.13.

Badi, I. & Bouraima, M. B. (2023) Development of MCDM-based frameworks for proactively managing the most critical risk factors for transport accidents: a case study in Libya. *Spectrum of Engineering and Management Sciences*. 1(1), 38-47. doi: 10.31181/ sems1120231b.

Baydaş, M., Kavacık, M. & Wang, Z. (2024) Interpreting the Determinants of Sensitivity in MCDM Methods with a New Perspective: An Application on E-Scooter Selection with the PROBID Method. *Spectrum of Engineering and Management Sciences*. 2(1), 17-35. doi: 10.31181/sems2120242b.

Chiu, A., Stumm, M. & Yuan, D. (2022) Investigating Managed Language Runtime Performance Why JavaScript and Python are 8x and 29x slower than C++, yet Java and Go can be faster? In*: Proceedings of the 2022 USENIX Annual Technical Conference (USENIX ATC 22), 11–13 July 2022, Carlsbad, CA, U.S.A*. Berkeley, California, U.S.A., USENIX – The Advanced Computing Systems Association. pp. 835-852.

Damjanović, M., Stević, Ž., Stanimirović, D., Tanackov, I. & Marinković, D. (2022) Impact of the number of vehicles on traffic safety: multiphase modeling. *Facta Universitatis, Series: Mechanical Engineering.* 20(1), 177-197. doi: 10.22190/ FUME220215012D.

Filip, F. G. (2021) Automation and Computers and Their Contribution to Human Well-being and Resilience. *Studies in Informatics and Control*. 30(4), 5-18. doi: 10.24846/v30i4y202101.

Hadžikadunic, A., Stevic, Ž., Badi, I. & Roso, V. (2023) Evaluating the Logistics Performance Index of European Union Countries: An Integrated Multi-Criteria Decision-Making Approach Utilizing the Bonferroni Operator. *International Journal of Knowledge and Innovation Studies*. 1(1), 44-59. doi: 10.56578/ijkis010104.

Jäger, D. & Gümmer, V. (2023) PythonDAQ – A Python based measurement data acquisition and processing software. In: *Proceedings of XXVI Biennial Symposium on Measuring Techniques in Turbomachinery (MTT2622)*, *28-30 September 2022, Pisa, Italy*. IOP Publishing. pp. 1-13. doi: 10.1088/1742-6596/2511/1/012032.

Kabassi, K., Karydis, C. & Botonis, A. (2020) AHP, Fuzzy SAW, and Fuzzy WPM for the Evaluation of Cultural Websites. *Multimodal Technologies and Interaction.* 4(1), 5. doi: 10.3390/mti4010005.

Katzy, J., Izadi, M. & Deursen, A. (2023) On the Impact of Language Selection for Training and Evaluating Programming Language Models. In: *Proceedings of 2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM) 02-03 October 2023*, *Bogotá, Colombia*. Piscataway, New Jersey, U.S.A., Institute of Electrical and Electronics Engineers (IEEE). pp. 271-276. doi: 10.1109/SCAM59687.2023.00038.

Keršuliene, V., Zavadskas, E. K. & Turskis, Z. (2010) Selection of rational dispute resolution method by applying new step-wise weight assessment ratio analysis (SWARA). *Journal of Business Economics and Management*. 11(2), 243-258. doi: 10.3846/ jbem.2010.12.

Keshavarz-Ghorabaee, M., Abdolghani, R., Maghsoud, A., Zavadskas, E. K. & Antuchevičienė, J. (2024) Multi-Criteria personnel evaluation and selection using an objective pairwise adjusted ratio analysis (OPARA). *Economic Computation and Economic Cybernetics Studies and Research*. 58(2), pp. 23-45. doi: 10.24818/18423264/58.2.24.02.

Li, X., Liao, H., Baušys, R. & Zavadskas, E. K. (2024) Large-scale emergency supplier selection considering limited rational behaviors of decision makers and ranking robustness. *Technological and Economic Development of Economy*. 30(4), 1037-1063. doi: 10.3846/tede.2024.21569.

Mishra, A. R., Chandel, A. & Motwani, D. (2020) Extended MABAC method based on discrimination measures for multi-criteria assessment of programming language with interval-valued intuitionistic fuzzy sets. *Granular Computing*. 5(1), pp. 97–117. doi: 10.1007/s41066-018-0130-5.

Naim, R., Nizam, M. F., Hanamasagar, S., Noureddine, J. & Miladinova, M. (2010) *Comparative Studies of 10 Programming Languages within 10 Diverse Criteria Revision 1.0.* Department of Computer Science and Software Engineering, Concordia University Montreal.

Padilla, J. R., Montefalcon, M. D. & Hernandez, A. (2023) Language AI in Programming: A Case Study of ChatGPT in Higher Education using Natural Language Processing. In: *Proceedings of 2023 11th IEEE Conference on Systems, Process & Control (ICSPC 2023)*, *16 December 2023, Malacca, Malaysia.* Piscataway, New Jersey, U.S.A., Institute of Electrical and Electronics Engineers (IEEE). pp. 1-7. doi: 10.1109/ICSPC59664.2023.10420194.

Parveen, Z. & Fatima, N. (2016) Performance Comparison of Most Common High Level Programming Languages. *International Journal of Computing Academic Research.* 5(5), 246-258.

Pejovic, P. (2019) Application of python programming language in measurements. *Facta Universitatis Series Electronics and Energetics*. 32(1), 1-23. doi: 10.2298/FUEE1901001P.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, P. & Saraiva, J. (2021) Ranking programming languages by energy efficiency. *Science of Computer Programming*. 205(6), 1-30. doi: 10.1016/j.scico.2021.102609.

Petrovas, A., Baušys, R. & Zavadskas, E. K. (2023) Gestalt Principles Governed Fitness Function for Genetic Pythagorean Neutrosophic WASPAS Game Scene Generation. *International Journal of Computers Communications & Control*. 18(4), 2-20. doi: 10.15837/ijccc.2023.4.5475.

Puška, A., Hodžić, I., Štilić, A. & Murtič, S. (2024) Evaluating European Union Countries on Climate Change Management: A Fuzzy MABAC Approach to the Climate Change Performance Index. *Journal of Green Economy and Low-Carbon Development.* 3(1), 15-25. doi: 10.56578/jgelcd030102.

Radovanović, M., Božanić, D., Tešić, D., Puška, A., Hezam, I. M. & Jana, C. (2023) Application of hybrid DIBR-FUCOM-LMAW-Bonferroni-grey-EDAS model in multicriteria decision-making. *Facta Universitatis, Series: Mechanical Engineering*. 21(3), pp. 387-403. doi: 10.22190/FUME230824036R.

Ristić, B., Bogdanović, V., Stević, Ž., Marinković, D., Papić, Z. & Gojković, P. (2024) Evaluation of Pedestrian Crossings Based on the Concept of Pedestrian Behavior Regarding Start-Up Time: Integrated Fuzzy MCDM Model. *Tehnički Vjesnik [Technical Gazette]*. 31(4), pp. 1206-1214. doi: 10.17559/TV-20240414001462.

Sakharkar, S. (2023) Systematic Review: Analysis of Coding Vulnerabilities across Languages. *Journal of Information Security*. 14(04), pp. 330-342. doi: 10.4236/jis.2023.144019.

Sambucci, L. & Paraschiv, E.A. (2024) The accelerated integration of artificial intelligence systems and its potential to expand the vulnerability of the critical infrastructure. Romanian Journal of Information Technology and Automatic Control, Vol. 34, No. 3, pp. 131-148.

Stein, M. & Geyer-Schulz, A. (2013) A Comparison of Five Programming Languages in a Graph Clustering Scenario. *Journal of Universal Computer Science*. 19(3), pp. 428-456. doi: 10.3217/jucs-019-03-0428.

Stević, Ž., Subotić, M., Softić, E. & Božić, B. (2022) Multi-Criteria Decision-Making Model for Evaluating Safety of Road Sections. *Journal of Intelligent Management Decision*. 1(2), pp. 78-87. doi: 10.56578/jimd010201.

Turskis, Z., Zavadskas, E. K., Antuchevičienė, J. & Kosareva, N. (2015) A hybrid model based on fuzzy AHP and fuzzy WASPAS for construction site selection. *International Journal of Computers Communications & Control*. 10(6), pp. 873-888. doi: 10.15837/ijccc.2015.6.2078.

Vrtagić, S., Softić, E., Subotić, M., Stević, Ž., Dordevic, M. & Ponjavic, M. (2021) Ranking Road Sections Based on MCDM Model: New Improved Fuzzy SWARA (IMF SWARA). *Axioms*. 10(2), pp.1-23. doi: 10.3390/axioms10020092.

Zakaria, A., Oualid H., Kaushik S. & Chitrang, P. (2015) *Comparative Studies of Six Programming Languages*. Cornell University, pp. 1-72.

Zakeri-Nasrabadia, M., Parsaa, S., Ramezania, M., Royb, C. & Ekhtiarzadeha, M. (2023) A systematic literature review on source code similarity measurement and clone detection: Techniques, applications, and challenges. *Journal of Systems and Software*. 204, 111796, pp. 1-49. doi: 10.1016/j.jss.2023.111796.

Zavadskas, E. K., Stanujkic, D., Karabasevic, D. & Turskis, Z. (2022) Analysis of the Simple WISP Method Results Using Different Normalization Procedures. *Studies in Informatics and Control*. 31(1), pp. 5-12. doi: 10.24846/v31i1y202201.