X

УДК 004.4:519.816 **DOI** 10.24412/2413-7383-2024-3-80-86

Р. С. Хакимов, Я. С. Пикалёв
Федеральное государственное бюджетное научное учреждение
«Институт проблем искусственного интеллекта», г. Донецк,
283048, г. Донецк, ул. Артема, 118 б

РАЗРАБОТКА МЕТОДИКИ ВЫБОРА ГРАФИЧЕСКОЙ ОБОЛОЧКИ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON НА ОСНОВЕ МЕТОДА АНАЛИЗА ИЕРАРХИЙ

R. S. Khakimov, Y. S. Pikaliov
Federal State Budgetary Scientific Institution «Institute of Artificial Intelligence Problems»
283048, Donetsk, Artema str, 118-b

DEVELOPMENT OF METHODOLOGY FOR SELECTION OF GRAPHICAL SHELL IN PYTHON PROGRAMMING LANGUAGE BASED ON HIERARCHICAL ANALYSIS METHOD

В статье рассмотрены ключевые моменты и особенности метода анализа иерархий, основные этапы применения метода: формирование иерархической структуры задачи, попарные сравнения критериев и альтернатив, проверка согласованности оценок и расчет глобальных приоритетов. В качестве критериев для оценки библиотек выделены функциональность, кроссплатформенность, ресурсоемкость и дизайн виджетов. Обосновано применение для выбора подходящей библиотеки на Python для создания графического интерфейса пользователя.
Ключевые слова: метод анализа иерархий, критерии, альтернативы, библиотека

The article discusses the key points and features of the hierarchy analysis method, the main stages of the method application: formation of the hierarchical structure of the task, pairwise comparisons of criteria and alternatives, checking the consistency of evaluations and calculation of global priorities. Functionality, cross-platform, resource intensity and widget design are identified as criteria for evaluating libraries. The application for selecting a suitable Python library for creating a graphical user interface is justified.
Key words: hierarchy analysis method, criteria, alternatives, library.

Введение

При решении определенных задач человек часто сталкивается с ситуацией, когда ему необходимо сделать выбор. Для помощи в принятии такого решения при нахождении подходящего варианта были изобретены различные математические методы. Данная работа посвящена методу анализа иерархий (МАИ, англ. Hierarchy Analysis Method) [1] и его применению для выбора подходящей библиотеки графического интерфейса пользователя (Graphical User Interface, GUI) для разработки приложений на Python.

Существует множество библиотек на языке Python, каждая из которых имеет свои особенности и преимущества. При этом выбор наиболее подходящей библиотеки может быть сложной задачей, учитывая наличие различных критериев и ограничений.

Целью данной работы является реализация метода выбора наилучшей библиотеки GUI на Python с использованием метода анализа иерархий. Будет рассмотрено, как МАИ может быть применен для формализации критериев выбора, сравнительного анализа библиотек и принятия решения о наиболее подходящей библиотеке для конкретного проекта.

Метод анализа иерархий – математический инструмент системного подхода к решению задач многокритериального выбора, который был разработан Т. Саати в 1970 г. для анализа сложных проблем и получения их решения с учетом всей имеющейся качественной и количественной информации, факторов, влияющих на ее решение, способов достижения целей и возможных альтернативных решений

МАИ не предписывает лицу, принимающему решение, какого-либо «правильного» решения, а позволяет ему в интерактивном режиме найти такой вариант (альтернативу), который наилучшим образом согласуется с его пониманием сути проблемы и требованиями к ее решению.

В МАИ рассматривается иерархия, состоящая из уровней критериев и альтернатив. На верхнем уровне располагаются цели принятия решений, которые декомпозируются на более низкие уровни критериев и альтернатив. Затем проводятся попарные сравнения для оценки важности критериев и альтернатив.

Основной инструмент МАИ – это матрица попарных сравнений. Для каждого уровня иерархии (критериев или альтернатив) строится квадратная матрица $A = [a_{ij}]$, где каждый элемент a_{ij} представляет собой результат попарного сравнения элемента i с элементом j. Шкала Саати используется для оценки, и обычно имеет значения от 1 до 9. В табл. 1 приведены оценки важности.

Таблица 1 – Шкала относительной важности

Интенсивность относительной важности	Определение	Объяснение
1	Равная важность	Равный вклад двух видов деятельности в цель
3	Умеренное превосходство одного над другим	Опыт и суждения дают легкое превосходство одного вида деятельности над другим
5	Существенное или сильное превосходство	Опыт и суждения дают сильное превосходство одного вида деятельности над другим

Продолж. табл. 1

7	Значительное превосходство	Одному виду деятельности дается настолько сильное превосходство, что оно становится практически значительным
9	Абсолютное превосходство	Превосходство одного над другим подтверждается наиболее сильно
2, 4, 6, 8	Промежуточное решение между двумя соседними суждениями	Применяются в компромиссном случае
Обратные величины приведенных выше чисел	Если при сравнении одного вида деятельности с другим получено одно из вышеуказанных чисел (например, 3), то при сравнении второго вида деятельности с первым получим обратную величину (т. е. 1/3)	

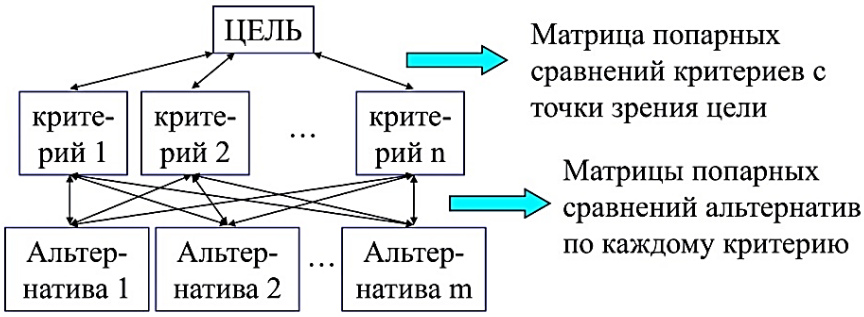

Рисунок 1 – Дерево критериев и альтернатив

Для метода анализа иерархий характерны следующие этапы:

1. Формирование иерархии целей. На первом этапе создается иерархическая структура задачи. Это позволяет разбить сложную проблему на более простые составляющие.

2. Определение приоритетов. На втором этапе проводится оценка значимости критериев и альтернатив относительно друг друга с использованием попарных сравнений. Это ключевой момент метода анализа иерархий.

3. Расчет локальных векторов приоритетов. На этом этапе производится расчет локальных векторов приоритетов для каждого уровня иерархии на основе данных из матрицы попарных сравнений.

Шаги:

– Нормализация матрицы попарных сравнений: Каждый элемент матрицы делится на сумму элементов своего столбца. Это приводит к нормализованной матрице, где сумма элементов каждого столбца равна 1.

– Вычисление локальных приоритетов: Для каждого элемента рассчитывается среднее арифметическое его строки в нормализованной матрице. Этот результат является локальным приоритетом (или весом) данного элемента относительно других.

4. Проверка экспертных оценок на непротиворечивость (вычисление индекса согласованности). Этот этап включает проверку согласованности (консистентности) попарных сравнений, чтобы убедиться, что они логичны и непротиворечивы.

Шаги:

– Вычисление максимального собственного значения матрицы (λ_{max}). Для этого матрица попарных сравнений умножается на вектор приоритетов, а результат делится на соответствующий элемент вектора приоритетов.

– Расчет индекса согласованности (*CI*):

$$CI = \frac{\lambda_{max} - n}{n - 1} \quad (1)$$

где *n* – размер матрицы.

– Расчет коэффициента согласованности (*CR*):

$$CR = \frac{CI}{RI} \quad (2)$$

где *RI* – случайный индекс согласованности (взятый из таблицы Саати).

– Проверка на согласованность: Если коэффициент согласованности CRCRCR меньше 0.1, оценки считаются достаточно согласованными. Если CRCRCR больше 0.1, необходимо пересмотреть попарные сравнения.

5. Расчет приоритетов целей и мероприятий для иерархии в целом на основе синтеза локальных приоритетов. На последнем этапе осуществляется синтез локальных приоритетов для получения глобальных весов (приоритетов) альтернатив. Вес каждой альтернативы умножается на вес соответствующего критерия. Затем эти значения складываются для каждой альтернативы по всем критериям.

$$\text{Глобальный приоритет альтернативы} = \sum_{i=1}^{m} \omega_i \cdot a_i \quad (3)$$

где ω_i – вес критерия, а a_i – локальный приоритет альтернативы по этому критерию. Альтернатива с наибольшим значением глобального приоритета является наилучшим выбором.

Что касается настоящего исследования, то для создания пользовательского интерфейса были рассмотрены следующие библиотеки: Kivy [2], TKinter [3], wxPython [4], DearPyGUI [5], PySimpleGUI [6], PySide [7]. С точки зрения МАИ, они будут являться альтернативами.

Для осуществления выбора был написан программный код в среде разработки PyCharm на языке Python 3.11. Пример результатов выполнения кода приведен на рис. 2.

```
Коэффициент степени отклонения: 6.130256002276839
Индекс согласованности: 0.02605120045536786
Отношение согласованности: 0.021009032625296662
Матрица векторов приоритетов альтернатив для эксперта #1:
[[0.06417041 0.10077435 0.04262183 0.16023749 0.25206167 0.38013425]
 [0.10077435 0.04262183 0.06417041 0.16023749 0.25206167 0.38013425]
 [0.04254634 0.06416414 0.10093344 0.16022185 0.38009714 0.25203707]
 [0.10077435 0.04262183 0.06417041 0.25206167 0.16023749 0.38013425]
 [0.06417041 0.10077435 0.04262183 0.25206167 0.16023749 0.38013425]
 [0.25170347 0.06418664 0.10096883 0.04263261 0.16027803 0.38023041]]
```

Рисунок 2 – Пример результатов выполнения кода на Python 3.11 в среде PyCharm

В качестве критериев оценки были выбраны следующие параметры: удобство, функциональность, скорость создания, современный дизайн виджетов, ресурсоемкость, кроссплатформенность.

Матрица сравнения критериев, созданная с помощью библиотеки Pandas на Python, согласно МАИ представлена на рис. 3.

	Удобство	Функцио-нальность	Скорость создания	Современный дизайн виджетов	Ресурсо-емкость	Кроссплат-форменность
Удобство	1,0	1/4	1/2	2,0	1/3	3,0
Функциональность	4,0	1,0	3,0	5,0	2,0	6,0
Скорость создания	2,0	1/3	1,0	3,0	1/2	4,0
Современный дизайн виджетов	1/2	1/5	1/3	1,0	1/4	2,0
Ресурсоемкость	3,0	1/2	2,0	4,0	1,0	5,0
Кроссплатформенность	1/3	1/6	1/4	1/2	1/5	1,0

Рисунок 3 – Матрица сравнения критериев для выбора GUI

Согласно МАИ, подобные матрицы сравнения выполняются для альтернатив по каждому критерию.

После составления матриц приоритетов проводится расчет локальных векторов и проверка оценок на непротиворечивость (вычисление индекса согласованности, ИС). Проверка рассогласованности позволяет выявить ошибки, которые мог допустить эксперт при заполнении матрицы парных сравнений.

$$ИС = (\lambda_{max} - n)/(n - 1) \quad (4)$$

где n – размерность матрицы, а λ_{max} вычисляется следующим образом:

– суммируется каждый столбец матрицы парных сравнений;

– сумма первого столбца умножается на первую компоненту локального вектора приоритетов, сумма второго столбца на вторую компоненту и т.д.;

– полученные произведения суммируются.

Если разделить ИС на число, соответствующее случайной согласованности (СС) матрицы того же порядка, получим отношение согласованности (ОС). Величина ОС должна быть порядка 10% или менее, чтобы быть приемлемой.

$$ОС = ИС / СС \quad (5)$$

Таблица 2 – Значения случайной согласованности (СС)

Размер матрицы	1	2	3	4	5	6	7	8	9	10
Случайная согласованность	0	0	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49

Если такие отклонения превышают установленные пределы, то следует перепроверить их в матрице. В данной работе оценка согласованности показала, что рейтинг приоритетов установлен корректно.

Заключение

После выполнения оставшихся расчетов результат МАИ показал, что согласно значению максимальной оценки: 0,36, наиболее подходящей библиотекой является: PySide, что говорит о том, что использование рассматриваемого метода позволяет систематизировать критерии выбора, оценить их важность и принять обоснованное решение о наиболее подходящей библиотеке для конкретного проекта.

Список литературы

1. Саати, Т. Л. *Принятие решений : Метод анализа иерархий* [Текст] / Т. Саати ; пер. с англ. Р. Г. Вачнадзе. Москва : Радио и связь, 1993. – 314 с.
2. *Kivy : official site* [Электронный ресурс]. Режим доступа : https://kivy.org/ (дата обращения: 04.04.2024).
3. *TKinter : official site [*Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/tkinter.html (дата обращения: 04.04.2024).
4. *wxPython : official site* [Электронный ресурс]. Режим доступа: https://wxpython.org (дата обращения: 05.04.2024).
5. *DearPyGUI : official site* [Электронный ресурс]. Режим доступа : https://dearpygui.readthedocs.io/en/latest/ index.html# (дата обращения: 05.04.2024).
6. *PySimpleGUI : official site*. [Электронный ресурс]. Режим доступа : https://pysimplegui.com (дата обращения: 06.04.2024).
7. *PySide : official site*. [Электронный ресурс]. Режим доступа : https://pyside.github.io (дата обращения: 06.04.2024).

References

1. Saati, T. L. Decision-making: Hierarchy analysis method [Text]/T. Saati; per. From the English R. G. Vachnadze. - Moscow: Radio and Communications, 1993. - 314 p.
2. Kivy: official site. - Access mode: https://kivy.org/ (date of access: 04.04.2024).
3. TKinter: official site. - Access mode: https://docs.python.org/3/library/tkinter.html (date of access: 04.04.2024).
4. wxPython: official site. - Access mode: https://wxpython.org (date of access: 05.04.2024).
5. DearPyGUI: official site. - Access mode: https://dearpygui.readthedocs.io/en/latest/ index.html # (date of access: 05.04.2024).
6. PySimpleGUI : official site. [Electronic resource]. - Access mode: https://pysimplegui.com (date of access: 06.04.2024).
7. PySide : official site. [Electronic resource]. - Access mode: https://pyside.github.io (date of access: 06.04.2024).

RESUME

R. S. Khakimov, Ya. S. Pikalev

Development Of Methodology For Selection Of Graphical Shell In Python Programming Language Based On Hierarchical Analysis Method

The article is devoted to the development of a methodology for selecting a library for creating a graphical user interface (GUI) in the Python programming language using the hierarchy analysis method (MAI). The main stages of the MAI application were considered: the formation of a hierarchical structure of the task, pairwise comparisons of criteria and alternatives, checking the consistency of assessments and calculating global priorities. Functionality, cross-platform, resource intensity and widget design are highlighted as criteria for evaluating libraries. As a result of analysis based on the hierarchy analysis method, the PySide library was chosen as the most suitable for creating GUI applications in Python.

РЕЗЮМЕ

Р. С. Хакимов, Я. С. Пикалёв
Разработка методики выбора графической оболочки на языке программирования PYTHON на основе метода анализа иерархий

Статья посвящена разработке методики выбора библиотеки для создания графического интерфейса пользователя (GUI) на языке программирования Python с использованием метода анализа иерархий (МАИ). Рассмотрены основные этапы применения МАИ: формирование иерархической структуры задачи, попарные сравнения критериев и альтернатив, проверка согласованности оценок и расчет глобальных приоритетов. В качестве критериев для оценки библиотек выделены функциональность, кроссплатформенность, ресурсоемкость и дизайн виджетов. В результате анализа на основе метода анализа иерархий была выбрана библиотека PySide как наиболее подходящая для создания GUI-приложений на Python.

Хакимов Ренат Саитович – младший научный сотрудник, Федеральное государственное бюджетное научное учреждение «Институт проблем искусственного интеллекта». *Область научных интересов:* компьютерное зрение, машинное обучение, нейронные сети, эл. почта khakimov.ru@mail.ru, адрес: 283048, г. Донецк, ул. Артема, д. 118 б, телефон +7949 4584995.

Пикалёв Ярослав Сергеевич – старший научный сотрудник, Федеральное государственное бюджетное научное учреждение «Институт проблем искусственного интеллекта», г. Донецк. *Область научных интересов:* Цифровая обработка сигналов, анализ данных, распознавание образов, обработка естественного языка, компьютерное зрение, машинное обучение, нейронные сети, эл. почта pikaliov.nlp@gmail.com, адрес: 283048, г. Донецк, ул. Артема, д. 118 б, телефон: +7949 4287388.

Статья поступила в редакцию 03.06.2024.