## Міністерство освіти України

# Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського" Факультет інформатики та обчислювальної техніки

## **3BIT**

до практикуму № 5

з дисципліни

"Інфомаційні технології підтримки прийняття рішень"

на тему: "Методи TOPSIS, VIKOR"

Варіант 11

Виконала:

Студентка групи ІП-71

Каспрук Анастасія Андріївна

## 1. Результати.

## (Нумерація альтернатив починається з нуля).

#### Частина 1.

- **А)** Найкраща альтернатива 14.
- **Б)** Найкраща альтернатива 14.

#### Частина 2.

- **2.1.** Компромісний розв'язок 14.
- **2.2.** Для всіх значень  $\vartheta$  [0; 1] компромісний розв'язок 14.

#### 2. Постановка задачі.

#### Завдання

Задано множину альтернатив {A1,...,A15}, що оцінені за критеріями k1..k12. Оцінки альтернатив за усіма критеріями дано в таблиці.

Кожен критерій має ваговий коефіцієнт wi (Таблиця із вхідними даними співпадає з даними для практикуму 4)

В методах TOPSIS і VIKOR застосовуються нормовані вагові коефіцієнти критеріїв, тому потрібно виконати нормування цих коефіцієнтів (згідно співвідношення (3) в описі методу VIKOR)

#### Необхідно:

#### Частина 1.

За допомогою метода TOPSIS встановити ранжування на множині альтернатив (згідно спадання важливості) та вибрати найкращу альтернативу для випадків:

- А) всі критерії потрібно максимізувати;
- Б) критерії k1-k7 підлягають максимізації, а критерії k8-k12 мінімізації (тобто  $k1-k7 \in K^+$ , а  $k8-k12 \in K^-$ ).

#### Частина 2.

- 1. За допомогою метода VIKOR встановити ранжування на множині альтернатив (згідно спаданню важливості) та вибрати найкращу альтернативу (компромісний розв'язок), якщо параметр  $\vartheta$ =0,5.Вважати, що всі критерії потрібно максимізувати.
- 2. Дослідити в методі VIKOR вплив зміни параметра  $\vartheta$  на результуючий компромісний розв'язок. Дослідити випадки  $0 \le \vartheta \le 1$  з кроком 0,1. В кожному випадку обгрунтувати результат ранжування і компромісний розв'язок.

Навести Висновки проведеного аналізу, а також виконати порівняння обох методів за стратегією та результатами.

#### Завдання для варіанту 11:

```
Таблиця оцінок альтернатив А1-А15 за критеріями к1-к12
3 8 9 7 7 10 7 4 5 4 2 6
3 6 7 4 6 9 9 10 7
                     3
       3 10 10 1 9 7 6 1 1
2 3 4 9 7 7 2 4 10 10
1 8 1 10 4 7 6 1 1
8 7 4 6 6
           2 1 10 6 10 3
10 8 6 5 5 3 1
                9 9
6 8 3 1 1 4 10 5 2
5 1 8 8 5 2 5 6 9
  3 4 3 10 2 4 6 9 7
       3 7 7
              6 5 1
  2 5
4 1 4 8 2
           8 2 7 3 7
4 10 2 3 1 5 3 9 1 10 4
4 8 6 5 2 6 2 8 2 6 3 8
4 9 7 9 7 9 2 9 8 10 2 6
Вагові коефіцієнти критеріїв к1-к12:
1 10 5 2 2 6 2 5 8 2 5 8
```

#### 3. Розв'язок.

#### Частина 1 А).

На наступному скріншоті відображено поетапний розв'язок задачі багатокритеріальної оптимізації методом TOPSIS для випадку, коли всі критерії потрібно максимізувати.

```
Microsoft Visual Studio Debug Console
                                                                                                                                                                                                           Поетапний розв'язок задачі методом TOPSIS
Нормалізовані оцінки альтернатив:
Зважені нормалізовані оцінки альтернатив:
PIS (утопічна точка): 0.009 0.072 0.039 0.015 0.015 0.041 0.019 0.032 0.059 0.013 0.042 0.064
NIS (антиутопічна точка): 0,001 0,007 0,004 0,001 0,002 0,008 0,002 0,003 0,006 0,001 0,005 0,008
D+ (відстань до PIS): 0,054 0,057 0,092 0,083 0,092 0,060 0,048 0,079 0,082 0,085 0,093 0,090 0,086 0,064 0,043
    (відстань до NIS): 0,086 0,077 0,059 0,065 0,059 0,085 0,090 0,062 0,075 0,059 0,049 0,052 0,074 0,084 0,096
С (наближеність до PIS):
(альтернатива: 14, наближеність: 0,691)
(альтернатива: 6, наближеність: 0,652)
(альтернатива: 6, наближеність: 0,652)
(альтернатива: 5, наближеність: 0,585)
(альтернатива: 1, наближеність: 0,585)
(альтернатива: 1, наближеність: 0,575)
(альтернатива: 13, наближеність: 0,569)
(альтернатива: 8, наближеність: 0,462)
(альтернатива: 12, наближеність: 0,462)
(альтернатива: 7, наближеність: 0,462)
(альтернатива: 7, наближеність: 0,440)
(альтернатива: 7, наближеність: 0,392)
(альтернатива: 4, наближеність: 0,390)
(альтернатива: 4, наближеність: 0,368)
(альтернатива: 11, наближеність: 0,368)
Найкраща альтернатива: 14
```

Спочатку були обчислені нормалізовані оцінки альтернатив за наступним співвідношенням.

$$r_{kj}(\mathbf{x}) = \frac{x_{kj}}{\sqrt{\sum_{k=1}^{n} x_{kj}^2}}, \quad k = 1,...,n; j = 1,...,m$$

Потім кожна нормалізована оцінка була домножена на вагу відповідного критерію.

$$v_{kj}(x) = w_j r_{kj}(x), k = 1,...,n; j = 1,...,m$$

Далі були обчислені координати до утопічної та антиутопічної точок PIS та NIS.

$$PIS = \mathbf{A}^{+} = \left\{ v_{1}^{+}(\mathbf{x}), v_{2}^{+}(\mathbf{x}), ..., v_{j}^{+}(\mathbf{x}), ..., v_{m}^{+}(\mathbf{x}) \right\}$$
(5)
$$= \left\{ \left( \max_{k} v_{kj}(\mathbf{x}) \mid j \in \mathcal{K}^{+} \right), \left( \min_{k} v_{kj} \mid j \in \mathcal{K}^{-} \right) \mid k = 1, ..., n \right\}$$

$$NIS = \mathbf{A}^{-} = \left\{ v_{1}^{-}(\mathbf{x}), v_{2}^{-}(\mathbf{x}), ..., v_{j}^{-}(\mathbf{x}), ..., v_{m}^{-}(\mathbf{x}) \right\}$$
(6)
$$= \left\{ \left( \min_{k} v_{kj}(\mathbf{x}) \mid j \in \mathcal{K}^{+} \right), \left( \max_{k} v_{kj}(\mathbf{x}) \mid j \in \mathcal{K}^{-} \right) \mid k = 1, ..., n \right\}$$

Наступним кроком стало обчислення відстаней кожної альтернативи до позитивної ідеальної точки PIS та негативної ідеальної точки NIS.

$$D_k^* = \sqrt{\sum_{j=1}^m \left[ v_{kj}(\mathbf{x}) - v_j^+(\mathbf{x}) \right]^2}, \quad k = 1,...,n$$

$$D_k^- = \sqrt{\sum_{j=1}^m \left[ v_{kj}(\mathbf{x}) - v_j^-(\mathbf{x}) \right]^2}, \quad k = 1,...,n$$

На базі значень  $D^*$  та  $D^*$  були встановлені наближеності кожної альтернативи до позитивної ідеальної точки PIS.

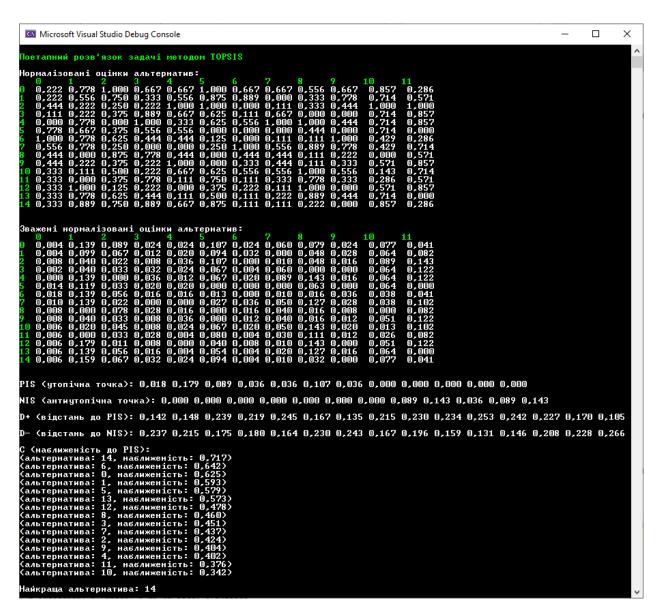
$$C_k^* = D_k^- / (D_k^* + D_k^-), \quad k = 1,...,n$$

де 
$$C_k^* \in [0,1] \quad \forall k = 1,...,n$$

Впорядкувавши значення  $C^*$  у порядку спадання, на першій позиції у множині отримали найбільш наближену до позитивної ідеальної точки PIS (тобто найкращу) альтернативу.

#### Частина 1 Б).

На наступному скріншоті відображено поетапний розв'язок задачі багатокритеріальної оптимізації методом TOPSIS для випадку, коли критерії k1-k7 підлягають максимізації, а критерії k8-k12 - мінімізації (тобто  $k1-k7 \in K^+$ , а  $k8-k12 \in K^-$ ).



Для цього випадку нормалізовані оцінки альтернатив обчислюються за наступними співвідношеннями.

1.1 Для критеріїв прибутку (
$$K^+$$
)
$$r_{kj}(x) = (x_{kj} - x_j^-)/(x_j^* - x_j^-),$$
де  $x_j^* = \max_k x_{kj}, x_j^- = \min_k x_{kj};$ 
або  $x_j^*$  – бажане значення,
 $x_j^-$  найгірше значення

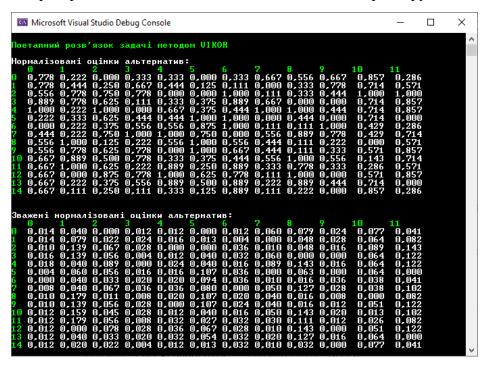
1.2 Для критеріїв витрат (
$$K^-$$
) 
$$r_{kj}\left(x\right) = (x^*_j - x_{kj})/(x^*_j - x^-_j),$$

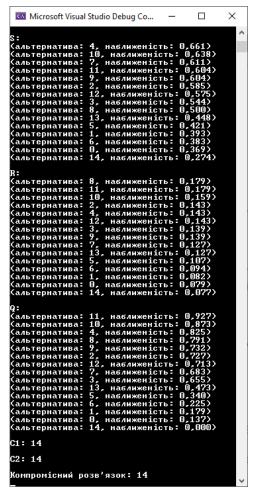
Всі інші співвідношення співпадають з тими, що наведені для розв'язку частини 1

A.

## Частина 2.1.

На наступних скріншотах відображено поетапний розв'язок задачі багатокритеріальної оптимізації методом VIKOR для параметру  $\vartheta$ =0,5.





Спочатку були обчислені нормалізовані оцінки альтернатив за наступним співвідношенням.

$$|f_{j}^{*}-f_{kj}|/|f_{j}^{*}-f_{j}^{-}|$$

Потім кожна нормалізована оцінка була домножена на вагу відповідного критерію.

$$w_j | f_j^* - f_{kj} | / | f_j^* - f_j^- |$$

Далі для кожної альтернативи були обчислені значення S та R за наступними співвідношеннями.

$$S_{k} = \sum_{j=1}^{m} w_{j} | f_{j}^{*} - f_{kj} | / | f_{j}^{*} - f_{j}^{-} |$$

$$R_{k} = \max_{j} \left\{ w_{j} | f_{j}^{*} - f_{kj} | / | f_{j}^{*} - f_{j}^{-} | \right.$$

$$j = 1, 2, ..., m$$

$$k = 1, 2, ..., n$$

 $S_k$  — визначає відстань від альтернативи  $A_k$  до компромісного розв'язку  $A^*$ , визначену за манхеттенською метрикою (чим менша відстань, тим ближче  $A_k$  до  $A^*$ ).

 $R_k$  — визначає відстань від альтернативи  $A_k$  до компромісного розв'язку  $A^*$ , визначену за метрикою Чебишева (чим менша відстань, тим ближче  $A_k$  до  $A^*$ ).

На базі значень S та R були обчислені значення Q.

$$Q_k = v(S_k - S^*)/(S^- - S^*) + \\ + (1-v)(R_k - R^*)/(R^- - R^*)$$
 де  $S^* = \min_k S_k$  (або  $S^* = 0$  - означає досягнення бажаного рівня)  $S^- = \max_k S_k$  (або  $S^- = 1$  - найгірший рівень),  $R^* = \min_k R_j$  (або  $R^* = 0$  і  $R^- = 1$  аналогічно)

На даному етапі маємо 3 ранжування альтернатив за значеннями S, R, Q. Впорядовуємо їх у порядку спадання.

Далі для відбору компромісного розв'язку необхідно перевірити 2 умови. Пропонується вважати компромісним розв'язком альтернативу а', яка в ранжуванні за Q є найкращою (мінімальне значення  $Q_k$ ), якщо виконуються дві наступні умови:

**С1.** «Прийнятна перевага»

**С2.** «Прийнятна стабільність».

## C1. «Прийнятна перевага»:

$$Q(a'') - Q(a') \ge DQ$$

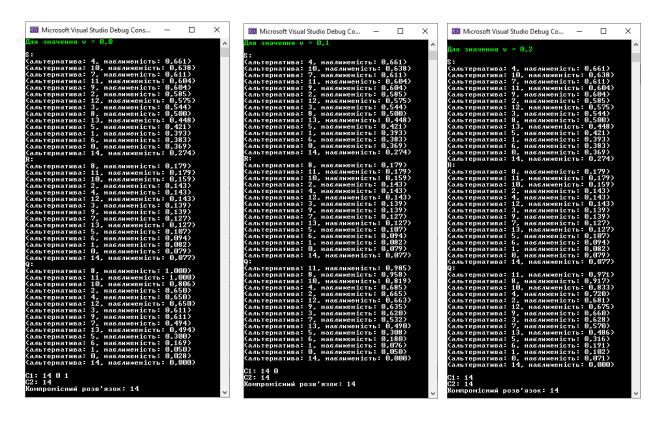
де a'' — наступна найкраща альтернатива в ранжуванні за Q після a', DQ = 1/(n-1), n — кількість альтернатив

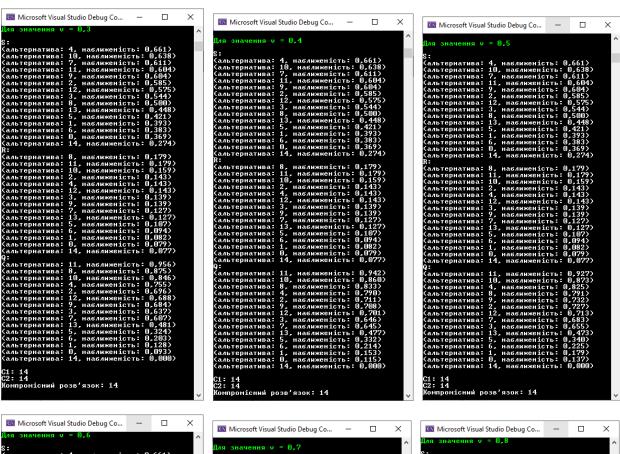
# **С2.** «Прийнятна стабільність»: Альтернатива a' повинна також мати і найкращі значення (мінімальні) в ранжуваннях за S та/або R

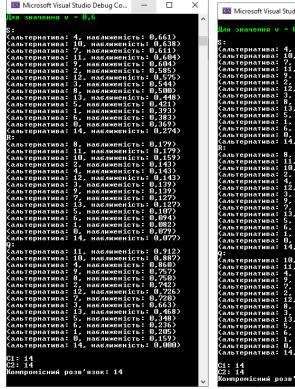
У компромісний розв'язок входять ті альтернативи, які задовольняють обидві умови.

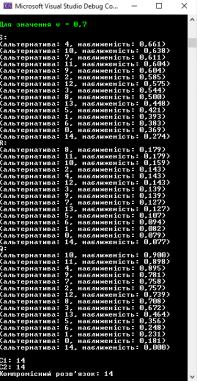
#### Частина 2.2.

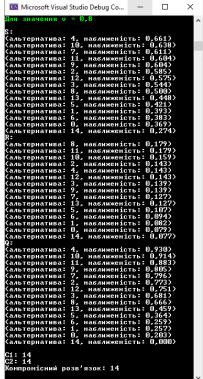
Дослідимо в методі VIKOR вплив зміни параметра  $\theta$  на результуючий компромісний розв'язок. Розглянемо випадки  $0 \le \theta \le 1$  з кроком 0,1.

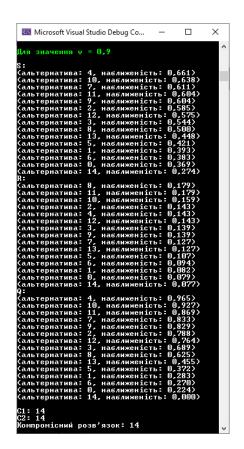


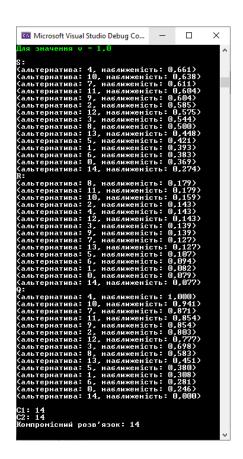












Бачимо, що зі зміною значення параметра  $\vartheta$  ранжування для Q також змінюється. Для малих значень  $\vartheta=0$  та  $\vartheta=0.1$  прийнятну перевагу задовольняє більша кількість альтернатив, а саме множини  $\{14,0,1\}$  та  $\{14,0\}$ . Однак, для більших значень параметру  $\vartheta$  прийнятну умову задовольняє лише одна альтернатива. Для всіх значень до компромісного розв'язку входить лише одна альтернатива, а саме 14.

## 4. Лістинг програми.

Посилання на github-репозиторій з кодом:

https://github.com/KasprukNastia/decisions/tree/master/Lab5

#### Клас TOPSIS

```
/// <summary>
  /// Клас, що описує метод TOPSIS
  /// </summary>
  public class TOPSIS
    /// <summary>
    /// Значення критеріїв для альтернатив
    /// </summary>
    public int[][] Evaluations { get; }
    /// <summary>
    /// Ваги критеріїв
    /// </summary>
    public IReadOnlyList<double> Weights { get; }
    /// <summary>
    /// Критерії, які потрібно максимізувати
    /// </summary>
    public IReadOnlyList<int> WeightsToMaximize { get; }
    /// <summary>
    /// Критерії, які потрібно мінімізувати
    /// </summary>
    public IReadOnlyList<int> WeightsToMinimize { get; }
    /// <summary>
    /// К-сть критеріїв
    /// </summary>
    public int CriteriasCount { get; }
    /// <summary>
    /// К-сть альтернатив
    /// </summary>
    public int AlternativesCount { get; }
    /// <summary>
    /// Нормалізовані оцінки альтернатив
    /// </summary>
    private double[][] _normalizedEvaluations;
    /// <summary>
    /// Нормалізовані оцінки альтернатив
    /// </summary>
    public double[][] NormalizedEvaluations
    {
      get
        if (_normalizedEvaluations == null)
          if (WeightsToMaximize == null && WeightsToMinimize == null)
            _normalizedEvaluations = CalcNormalizedAllMax();
            _normalizedEvaluations = CalcNormalizedMinAndMax();
        return _normalizedEvaluations;
      }
    }
```

```
/// <summary>
/// Зважені нормалізовані оцінки альтернатив
/// </summary>
private double[][] _weightedNormalizedEvaluations;
/// <summary>
/// Зважені нормалізовані оцінки альтернатив
/// </summary>
public double[][] WeightedNormalizedEvaluations
  get
    if (_weightedNormalizedEvaluations == null)
       weightedNormalizedEvaluations = new double[AlternativesCount][];
      for (int i = 0; i < AlternativesCount; i++)
         weightedNormalizedEvaluations[i] = new double[CriteriasCount];
        for (int j = 0; j < CriteriasCount; j++)
           _weightedNormalizedEvaluations[i][j] = NormalizedEvaluations[i][j] * Weights[j];
        }
      }
    }
    return _weightedNormalizedEvaluations;
 }
}
/// <summary>
/// Утопічна точка
/// </summary>
private IReadOnlyList<double>_pis;
/// <summary>
/// Утопічна точка
/// </summary>
public IReadOnlyList<double> PIS
  get
    if (_pis == null)
      if (WeightsToMaximize == null && WeightsToMinimize == null)
        _pis = Enumerable.Range(0, CriteriasCount)
           . Select (criteria => GetMinAndMaxForCriteriaInWeighted (criteria)) \\
           .Select(minMax => minMax.max).ToList();
      else
        _pis = Enumerable.Range(0, CriteriasCount)
           .Select(criteria => GetMinAndMaxForCriteriaInWeighted(criteria))
           .Select((minMax, criteria) => WeightsToMaximize.Contains(criteria) ? minMax.max : minMax.min)
           .ToList();
    }
    return _pis;
}
/// <summary>
/// Антиутопічна точка
/// </summary>
private IReadOnlyList<double> _nis;
/// <summary>
/// Антиутопічна точка
/// </summary>
public IReadOnlyList<double> NIS
{
  get
    if(_nis == null)
```

```
{
      if (WeightsToMaximize == null && WeightsToMinimize == null)
         nis = Enumerable.Range(0, CriteriasCount)
           .Select(criteria => GetMinAndMaxForCriteriaInWeighted(criteria))
           .Select(minMax => minMax.min).ToList();
      else
         _nis = Enumerable.Range(0, CriteriasCount)
           .Select(criteria => GetMinAndMaxForCriteriaInWeighted(criteria))
           .Select((minMax, criteria) => WeightsToMaximize.Contains(criteria) ? minMax.min : minMax.max)
           .ToList();
    }
    return _nis;
  }
}
/// <summary>
/// Відстані до утопічної точки
/// </summary>
private IReadOnlyList<double>_dToPis;
/// <summary>
/// Відстані до утопічної точки
/// </summary>
public IReadOnlyList<double> DToPIS
{
  get
    if(_dToPis == null)
      double[] dToPis = new double[AlternativesCount];
      for(int alternative = 0; alternative < AlternativesCount; alternative++)</pre>
        dToPis[alternative] = Math.Sqrt(
           WeightedNormalizedEvaluations[alternative]
             .Select((elem, criteria) => Math.Pow(elem - PIS[criteria], 2))
             .Sum());
      _dToPis = dToPis.ToList();
    return _dToPis;
}
/// <summary>
/// Відстані до антиутопічної точки
/// </summary>
private IReadOnlyList<double> _dToNis;
/// <summary>
/// Відстані до антиутопічної точки
/// </summary>
public IReadOnlyList<double> DToNIS
{
  get
    if (_dToNis == null)
      double[] dToNis = new double[AlternativesCount];
      for (int alternative = 0; alternative < AlternativesCount; alternative++)
        dToNis[alternative] = Math.Sqrt(
           WeightedNormalizedEvaluations[alternative]
             .Select((elem, criteria) => Math.Pow(elem - NIS[criteria], 2))
             .Sum());
       dToNis = dToNis.ToList();
    return _dToNis;
```

```
}
/// <summary>
/// Наближеності до утопічної точки
/// </summary>
private IReadOnlyList<(int alternative, double closennes)> _c;
/// <summary>
/// Наближеності до утопічної точки
/// </summary>
public IReadOnlyList<(int alternative, double closennes)> C
  get
    if( c == null)
      _c = DToNIS.Select((dToNis, alternative) => (alternative, dToNis / (DToPIS[alternative] + dToNis)))
         .OrderByDescending(elem => elem.Item2)
         .ToList();
    return _c;
  }
}
public TOPSIS(int[][] evaluations,
  IReadOnlyList<double> weights,
  IReadOnlyList<int> weightsToMaximize = null,
  IReadOnlyList<int> weightsToMinimize = null)
{
  Evaluations = evaluations ?? throw new ArgumentNullException(nameof(evaluations));
  AlternativesCount = evaluations.Length;
  if (AlternativesCount > 0)
    CriteriasCount = evaluations[0].Length;
  for (int i = 1; i < AlternativesCount; i++)
    if (evaluations[i].Length != CriteriasCount)
      throw new ArgumentException($"Evaluation must be provided only for {CriteriasCount} criterias");
  Weights = weights ?? throw new ArgumentNullException(nameof(weights));
  if (Weights.Count != CriteriasCount)
    throw new ArgumentException("The number of weights does not meet the number of criterias");
  double weightsSum = Weights.Sum();
  Weights = Weights.Select(w => w / weightsSum).ToList();
  WeightsToMaximize = weightsToMaximize;
  WeightsToMinimize = weightsToMinimize;
}
/// <summary>
/// Обчислення нормалізованих оцінок альтернатив для випадку, коли всі критерії потрібно максимізувати
/// </summary>
private double[][] CalcNormalizedAllMax()
  double[][] normalized = new double[AlternativesCount][];
  List<double> geomMeansForCriterias = Enumerable.Range(0, CriteriasCount)
    .Select(criteria => CalcGeometricMeanForCriteria(criteria)).ToList();
  for (int i = 0; i < AlternativesCount; i++)
    normalized[i] = new double[CriteriasCount];
    for(int j = 0; j < CriteriasCount; j++)</pre>
      normalized[i][j] = Evaluations[i][j] / geomMeansForCriterias[j];
  }
```

```
return normalized:
}
/// <summary>
/// Обчислення нормалізованих оцінок альтернатив для випадку,
/// коли частину критеріїв потрібно максимізувати, а частину мінімізувати
/// </summary>
private double[][] CalcNormalizedMinAndMax()
  double[][] normalized = new double[AlternativesCount][];
  List<(int min, int max)> minMaxForCriterias = Enumerable.Range(0, CriteriasCount)
    .Select(criteria => GetMinAndMaxForCriteria(criteria)).ToList();
  for (int i = 0; i < AlternativesCount; i++)
    normalized[i] = new double[CriteriasCount];
    for (int j = 0; j < CriteriasCount; j++)
      (int min, int max) = minMaxForCriterias[j];
      normalized[i][j] = WeightsToMaximize.Contains(j)?
        (Evaluations[i][j] - min) / (double)(max - min) :
        (max - Evaluations[i][j]) / (double)(max - min);
    }
  }
  return normalized;
/// <summary>
/// Обчислення середнього геометричного для значень критерію
/// </summary>
private double CalcGeometricMeanForCriteria(int criteria)
  double sum = 0;
  for(int alternative = 0; alternative < AlternativesCount; alternative++)
    sum += Math.Pow(Evaluations[alternative][criteria], 2);
  }
  return Math.Sqrt(sum);
/// <summary>
/// Обчислення мінімального та максимально значень для критерію
/// з початкових оцінок альтернатив
/// </summary>
private (int min, int max) GetMinAndMaxForCriteria(int criteria)
  int[] criteriaValues = new int[AlternativesCount];
  for (int i = 0; i < AlternativesCount; i++)
    criteriaValues[i] = Evaluations[i][criteria];
  return (criteriaValues.Min(), criteriaValues.Max());
/// <summary>
/// Обчислення мінімального та максимально значень для критерію
/// зі зважених оцінок альтернатив
/// </summary>
private (double min, double max) GetMinAndMaxForCriteriaInWeighted(int criteria)
  double[] criteriaValues = new double[AlternativesCount];
  for (int i = 0; i < AlternativesCount; i++)
    criteriaValues[i] = WeightedNormalizedEvaluations[i][criteria];
```

```
}
      return (criteriaValues.Min(), criteriaValues.Max());
     Клас VIKOR
/// <summary>
  /// Клас, що описує метод VIKOR
  /// </summary>
  public class VIKOR
    /// <summary>
    /// Значення критеріїв для альтернатив
    /// </summary>
    public int[][] Evaluations { get; }
    /// <summary>
    /// Значення коефіцієнту V
    /// </summary>
    public double VCoef { get; }
    /// <summary>
    /// Ваги критеріїв
    /// </summary>
    public IReadOnlyList<double> Weights { get; }
    /// <summary>
    /// К-сть критеріїв
    /// </summary>
    public int CriteriasCount { get; }
    /// <summary>
    /// К-сть альтернатив
    /// </summary>
    public int AlternativesCount { get; }
    /// <summary>
    /// Нормалізовані оцінки альтернатив
    /// </summary>
    private double[][] _normalizedEvaluations;
    /// <summary>
    /// Нормалізовані оцінки альтернатив
    /// </summary>
    public double[][] NormalizedEvaluations
      get
        if (_normalizedEvaluations == null)
          _normalizedEvaluations = new double[AlternativesCount][];
           IReadOnlyList<(int min, int max)> minMaxForCriterias = Enumerable.Range(0, CriteriasCount)
             .Select(criteria => GetMinAndMaxForCriteria(criteria)).ToList();
           for (int i = 0; i < AlternativesCount; i++)
             _normalizedEvaluations[i] = new double[CriteriasCount];
             for (int j = 0; j < CriteriasCount; j++)</pre>
               (int min, int max) = minMaxForCriterias[j];
               \_normalized Evaluations[i][j] = (max - Evaluations[i][j]) / (double)(max - min);
          }
```

```
return normalizedEvaluations;
  }
}
/// <summary>
/// Зважені нормалізовані оцінки альтернатив
/// </summary>
private double[][] _weightedNormalizedEvaluations;
/// <summary>
/// Зважені нормалізовані оцінки альтернатив
/// </summary>
public double[][] WeightedNormalizedEvaluations
  get
    if ( weightedNormalizedEvaluations == null)
       _weightedNormalizedEvaluations = new double[AlternativesCount][];
      for (int i = 0; i < AlternativesCount; i++)
         _weightedNormalizedEvaluations[i] = new double[CriteriasCount];
        for (int j = 0; j < CriteriasCount; j++)
          _weightedNormalizedEvaluations[i][j] = NormalizedEvaluations[i][j] * Weights[j];
        }
      }
    }
    return weightedNormalizedEvaluations;
}
/// <summary>
/// Значення S - середні інтервали покращення
/// </summary>
private IReadOnlyList<(int alternative, double sValue)> _s;
/// <summary>
/// Значення S - середні інтервали покращення
/// </summary>
public IReadOnlyList<(int alternative, double sValue)> S
{
  get
    if(_s == null)
      _s = WeightedNormalizedEvaluations.Select((wne, alternative) => (alternative, wne.Sum()))
        .ToList();
    return _s;
  }
}
/// <summary>
/// Значення R - максимальні інтервали покращення
/// </summary>
private IReadOnlyList<(int alternative, double rValue)> _r;
/// <summary>
/// Значення R - максимальні інтервали покращення
/// </summary>
public IReadOnlyList<(int alternative, double rValue)> R
  get
    if (r == null)
      _r = WeightedNormalizedEvaluations.Select((wne, alternative) => (alternative, wne.Max()))
        .ToList();
    return r;
  }
}
```

```
/// <summary>
    /// Значення Q
    /// </summary>
    private IReadOnlyList<(int alternative, double qValue)> q;
    /// <summary>
    /// Значення Q
    /// </summary>
    public IReadOnlyList<(int alternative, double qValue)> Q
    {
      get
        if(q == null)
           var sValues = S.Select(pair => pair.sValue);
           var rValues = R.Select(pair => pair.rValue);
           double sMin = sValues.Min(), sMax = sValues.Max(), rMin = rValues.Min(), rMax = rValues.Max();
           _q = Enumerable.Range(0, AlternativesCount)
             .Select(alternative =>
               (alternative, (VCoef * (S[alternative].sValue - sMin) / (sMax - sMin)) + ((1 - VCoef) * (R[alternative].rValue - rMin)
/ (rMax - rMin))))
             .ToList();
        return _q;
      }
    }
    /// <summary>
    /// Значення С1 - прийнята перевага
    /// </summary>
    private List<int>_c1;
    /// <summary>
    /// Значення С1 - прийнята перевага
    /// </summary>
    public IReadOnlyList<int> C1
      get
        if(_c1 == null)
           IReadOnlyList<(int alternative, double qValue)> orderedQ = Q.OrderBy(elem => elem.qValue).ToList();
           _c1 = new List<int>();
           _c1.Add(orderedQ.First().alternative);
           for(int i = 1; i < AlternativesCount; i++)</pre>
             if ((orderedQ[i].qValue - orderedQ.First().qValue) < (1 / (double)(AlternativesCount - 1)))
               _c1.Add(orderedQ[i].alternative);
             else break;
           }
        }
        return c1;
    /// <summary>
    /// Значення С2 - прийнята стабільність
    /// </summary>
    private List<int> _c2;
    /// <summary>
    /// Значення С2 - прийнята стабільність
    /// </summary>
    public IReadOnlyList<int> C2
      get
```

```
if (_c2 == null)
    {
      IReadOnlyList<(int alternative, double sValue)> orderedS = S.OrderBy(elem => elem.sValue).ToList();
      IReadOnlyList<(int alternative, double rValue)> orderedR = R.OrderBy(elem => elem.rValue).ToList();
       _c2 = new List<int>();
      foreach(int alternative in C1)
        if (S[alternative].sValue == orderedS.First().sValue | |
           R[alternative].rValue == orderedR.First().rValue)
           _c2.Add(alternative);
    }
    return c2;
  }
}
/// <summary>
/// Фінальний результат, що задовольняє обидві умови С1 і С2
/// </summary>
private IReadOnlyList<int>_finalResult;
/// <summary>
/// Фінальний результат, що задовольняє обидві умови С1 і С2
/// </summary>
public IReadOnlyList<int> FinalResult
{
  get
    if( finalResult == null)
       _finalResult = C1.Intersect(C2).ToList();
    return _finalResult;
}
public VIKOR(int[][] evaluations,
  double vCoef.
  IReadOnlyList<double> weights)
  Evaluations = evaluations ?? throw new ArgumentNullException(nameof(evaluations));
  if (vCoef < 0 | | vCoef > 1)
    throw new ArgumentException($"{nameof(vCoef)} value must be value between 0 and 1");
  VCoef = vCoef;
  AlternativesCount = evaluations.Length;
  if (AlternativesCount > 0)
    CriteriasCount = evaluations[0].Length;
  for (int i = 1; i < AlternativesCount; i++)
    if (evaluations[i].Length != CriteriasCount)
      throw new ArgumentException($"Evaluation must be provided only for {CriteriasCount} criterias");
  Weights = weights ?? throw new ArgumentNullException(nameof(weights));
  if (Weights.Count != CriteriasCount)
    throw new ArgumentException("The number of weights does not meet the number of criterias");
  double weightsSum = Weights.Sum();
  Weights = Weights.Select(w => w / weightsSum).ToList();
}
/// <summary>
/// Обчислення мінімального та максимально значень для критерію
/// з початкових оцінок альтернатив
/// </summary>
private (int min, int max) GetMinAndMaxForCriteria(int criteria)
```

```
int[] criteriaValues = new int[AlternativesCount];
for (int i = 0; i < AlternativesCount; i++)
{
      criteriaValues[i] = Evaluations[i][criteria];
}
return (criteriaValues.Min(), criteriaValues.Max());
}
</pre>
```

# 5. Опис класів. Перелік розроблених функцій на методів.

Клас	Властивість	Опис	Тип значення, що повертає
TOPSIS	Evaluations	Значення критеріїв для альтернатив	int[][]
	Weights	Ваги критеріїв	IReadOnlyCollection <double></double>
	WeightsToMaximiz e	Критерії, які потрібно максимізувати	IReadOnlyList <int></int>
	WeightsToMinimiz e	Критерії, які потрібно мінімізувати	IReadOnlyList <int></int>
	CriteriasCount	К-сть критеріїв	int
	AlternativesCount	К-сть альтернатив	int
	NormalizedEvaluati ons	Нормалізовані оцінки альтернатив	double[][]
	WeightedNormalize dEvaluations	Зважені нормалізовані оцінки альтернатив	double[][]
	PIS	Утопічна точка	IReadOnlyList <doub le=""></doub>
	NIS	Антиутопічна точка	IReadOnlyList <doub le=""></doub>
	DToPIS	Відстані до утопічної точки	IReadOnlyList <doub le=""></doub>
	DToNIS	Відстані до антиутопічної точки	IReadOnlyList <doub le=""></doub>
	С	Наближеності до утопічної точки	IReadOnlyList<(int alternative, double closennes)>
VIKOR	Evaluations	Значення критеріїв для альтернатив	int[][]
	VCoef	Значення коефіцієнту V	double
	Weights	Ваги критеріїв	IReadOnlyCollection <double></double>
	CriteriasCount	К-сть критеріїв	int
	AlternativesCount	К-сть альтернатив	int
	NormalizedEvaluati ons	Нормалізовані оцінки альтернатив	double[][]
	WeightedNormalize dEvaluations	Зважені нормалізовані оцінки альтернатив	double[][]
	S	Значення S - середні інтервали покращення	IReadOnlyList<(int alternative, double sValue)>
	R	Значення R - максимальні інтервали покращення	IReadOnlyList<(int alternative, double rValue)>
	Q	Значення Q	IReadOnlyList<(int alternative, double qValue)>
	C1	Значення С1 - прийнята перевага	IReadOnlyList <int></int>
	C2	Значення С2 - прийнята стабільність	IReadOnlyList <int></int>
	FinalResult	Фінальний результат, що задовольняє обидві умови C1 і C2	IReadOnlyList <int></int>

Клас	Ф-ція/Метод	Параметри	Опис	Значення,
				що поверта€
TOPSIS	CalcNormalized	-	Обчислення нормалізованих	double[][] -
	AllMax		оцінок альтернатив для	нормалізовані
			випадку, коли всі критерії	оцінки
			потрібно максимізувати	альтернатив
	CalcNormalized	-	Обчислення нормалізованих	double[][] -
	MinAndMax		оцінок альтернатив для	нормалізовані
			випадку, коли частину	оцінки
			критеріїв потрібно максимізувати, а частину	альтернатив
			максимізувати, а частину мінімізувати	
	CalcGeometric	int criteria –	Обчислення середнього	double –
	MeanForCriteria	номер критерію	геометричного для значень	середн€
	Tyledin of Chieffa	помер критерио	критерію	геометричне
			nppe	для значень
l				критерію
	GetMinAndMax	int criteria –	Обчислення мінімального та	int min –
	ForCriteria	номер критерію	максимально значень для	мінімальне
			критерію з початкових оцінок	значення
			альтернатив	критерію;
				int max –
				максимальне
				значення
	G 16 4 126		0.5	критерію
	GetMinAndMax	int criteria –	Обчислення мінімального та	int min –
	ForCriteriaInWe	номер критерію	максимально значень для	мінімальне
	ighted		критерію зі зважених оцінок	значення
			альтернатив	критерію; int max –
				int max – максимальне
				значення
				критерію
VIKOR	GetMinAndMax	int criteria –	Обчислення мінімального та	int min –
	ForCriteria		максимально значень для	мінімальне
			критерію з початкових оцінок	значення
			альтернатив	критерію;
				int max –
				максимальне
				значення
				критерію

#### 6. Висновки.

У даній лабораторній роботі ми мали можливість навчитися розв'язувати задачі багатокритеріальної опитимізації застосовуючи методи TOPSIS і VIKOR.

Основна ідея методу TOPSIS дуже проста: після визначення «ідеального» і «ідеально-негативного» очікуваного станів проводиться спроба пошуку такого рішення, яке б дозволяло максимально наблизитися до «ідеального» і залишатися максимально віддаленим від «ідеально-негативного».

Процес прийняття рішення починається з раунду оцінювання, коли експерт оцінює всі альтернативні рішення за всіма критеріями. В результаті формується матриця рішень.

Metoд TOPSIS складається з 6 послідовних кроків:

- 1) розрахунок нормалізованої матриці рішення;
- 2) розрахунок зваженої нормалізованої матриці рішення;
- 3) визначення «ідеального» і «ідеально-негативного» очікуваного станів;
- 4) розрахунок віддаленості альтернатив до «ідеального» і «ідеальнонегативного» станів;
- 5) розрахунок відносній близькості до «ідеального» стану;
- 6) ранжування критеріїв.

Метод VIKOR також базується на визначені віддаленості альтернативи до компромісного розв'язку  $A^*$ . Отримане цим методом рішення може бути прийнято за найкраще, оскільки воно забезпечує максимальну корисність більшості (представлену min S) і мінімальні індивідуальні втрати опонента (представлене min R). Параметри S і R інтегровані в Q для визначення компромісного рішення, що базується на взаємних поступках. Коефіцієнт  $\vartheta$  дозволяє впливати на це компромісне рішення, з наданням переваги S або R.

Обидва методи дозволяють визначити найкращу альтернативу. Метод TOPSIS зрозуміліший для реалізації, однак VIKOR гнучкіший.

У результаті ми навчилися розв'язувати задачі багатокритеріальної опитимізації застосовуючи методи TOPSIS і VIKOR та змогли знайти рішення задачі у відповідності з варіантом.