**STEP BY STEP CODING IN C# OF FUZZY ALGORITHM**

**The Development: Automated Fuzzy Software Tool implemented in an Object Oriented Programming Language**

The approach used in this section is as follows:

1. Each step of the fuzzy TOPSIS algorithm is delineated in words.
2. The mathematical equations for the step are provided,
3. The program implementation for the step is written as a program segment in an object oriented language namely, C#. Step 2 and Step 5 references a utility class with helper methods.

***Step 1: Determine the weighting of Evaluation Criteria and weighting of Alternatives. Convert linguistic code to its corresponding fuzzy number***

Linguistic scales for evaluation criteria and alternatives are represented as a triangular fuzzy number. The fuzzy ratings for the criteria weights and alternative weights by expert decision makers have been preset in the web application as shown in the Table 1. Expert decision makers must choose the appropriate linguistic variables for criteria and the appropriate linguistic variables for alternatives with respect to the criteria.

|  |  |  |
| --- | --- | --- |
| **Criteria Weights** | **Fuzzy Number** | **Alternative Weights** |
| Very Low (VL) | (1,1,3) | Very weak (VW) |
| Low (L) | (1,3,5) | Weak (W) |
| Medium (M) | (3,5,7) | Average (A) |
| High (H) | (5,7,9) | Good (G) |
| Very High (VH) | (7,9,9) | Very Good (VG) |

**Table 1: Fuzzy Ratings for the criteria and alternatives**

The code below converts the linguistic weight to its corresponding fuzzy weight.

public class Step1

    {

       /// <summary>

/// To Convert a linguistic rating to its corresponding FuzzyNumber </summary>

/// <param name="code"> Linguistic term code </param>

/// <returns> The fuzzy.FuzzyNumber corresponding to the linguistic code </returns>

private FuzzyNumber[] \_fuzzyNumbers;

public Step1()

        {

            \_fuzzyNumbers = new FuzzyNumber[5];

            \_fuzzyNumbers[0] = new FuzzyNumber(1, 1, 3);    // Very Weak or Very Low

            \_fuzzyNumbers[1] = new FuzzyNumber(1, 3, 5);    // Weak or Low

            \_fuzzyNumbers[2] = new FuzzyNumber(3, 5, 7);    // Average or Medium

            \_fuzzyNumbers[3] = new FuzzyNumber(5, 7, 9);    // Good or High

            \_fuzzyNumbers[4] = new FuzzyNumber(7, 9, 9);    // Very Good or Very High

        }

        public FuzzyNumber LinguisticToFuzzy(string code)

        {

             // Very Poor or Very Low

            if (code == "Very Weak" || code == "Very Low")

            {

                return \_fuzzyNumbers[0];

            }

            // Poor or Low

            else if (code == "Weak" || code == "Low")

            {

                return \_fuzzyNumbers[1];

            }

            // Average or Medium

            else if (code == "Average" || code == "Medium")

            {

                return \_fuzzyNumbers[2];

            }

            // Good or High

            else if (code == "Good" || code == "High")

            {

                return \_fuzzyNumbers[3];

            }

            // Very Good or Very High

            else if (code == "Very Good" || code == "Very High")

            {

                return \_fuzzyNumbers[4];

            }

            throw new System.ArgumentException("Invalid linguistic code used");

        }

    }

***Step 2: Construct the aggregate weights vector and aggregate fuzzy decision matrix***

1. Mathematically aggregate weights of each criterion as follows:

*Let K = number of decision makers*

*Let j = 1,2, ….n representing the criteria*

= (

where: , (1)

1. Mathematically aggregate weights of each alternative as follows:

*Let k = number of decision makers*

*Let j = 1,2, ….n represent the criteria*

*Let i = 1,2,….s represent the alternatives*

Aggregate weightings of alternatives (*i*) with respect to each criterion *(j)* based on the fuzzy ratings by the *k* decision maker is expressed by the calculations below:

, (2)

1. Mathematically the aggregate fuzzy decision matrix for the alternatives (is expressed below:

C1  C2  Cj  Cm

(3)

The code for Step 2 is provided below.

public class Step2

{

Utils \_utils = new Utils();

/// <summary>

/// Calculates the alternatives aggregate fuzzy values.

/// </summary>

/// <returns>The alternatives agg fuzzy values.</returns>

/// <param name="dMRatings">DMRatings.</param>

public FuzzyNumber[] CalcAlternativesAggFuzzyValues(string[][] dMRatings)

        {

            FuzzyNumber[] fuzzyValues = new FuzzyNumber[dMRatings[0].Length];

            string[] iCriteriaRatings = new string[dMRatings.Length]; // the criteria ratings as per number of DMs

            for (int i = 0; i < dMRatings[0].Length; i++)

            {

                for (int j = 0; j < dMRatings.Length; j++)

                {

                    iCriteriaRatings[j] = dMRatings[j][i];

                }

                fuzzyValues[i] = \_utils.QualitativeToFuzzy(iCriteriaRatings);

            }

            return fuzzyValues;

        }

    }

***Step 3: Normalise the Fuzzy decision matrix***

*Let = normalized decision matrix*

*Let j = 1,2, ….n represent the criteria*

*Let i = 1,2,….s represent the alternatives*

*Let ( = aggregated weighting of alternative (i) with respect to criterion (j)*

*Let = triangular fuzzy number of alternative (i) with respect to criterion (j) in matrix r with n rows and s columns*

The normalized decision matrix is calculated as follows:

*where*  and = maxi  (benefit criteria) (4)

therefore = maxi  (9,9,9)

The code for the normalisation (Step 3) is provided below.

public class Step3

{

/// <summary>

/// Normalize an array of fuzzy numbers

/// </summary>

/// <param name="fuzzyValues"></param>

/// <returns></returns>

public FuzzyNumber[] NormalizeFuzzyValues(FuzzyNumber[] fuzzyValues)

{

FuzzyNumber[] normalizedArray = new FuzzyNumber[fuzzyValues.Length];

double[] maxArray = new double[fuzzyValues.Length];

double iMax;

// calculate maximum upper value

for (int j = 0; j < fuzzyValues.Length; j++)

{

maxArray[j] = fuzzyValues[j].Max;

}

// get the highest upper value

Array.Sort(maxArray);

iMax = maxArray[maxArray.Length - 1];

for (int i = 0; i < fuzzyValues.Length; i++)

{

normalizedArray[i] = new FuzzyNumber(fuzzyValues[i].Min / iMax, fuzzyValues[i].Mean / iMax,

fuzzyValues[i].Max / iMax);

}

return normalizedArray;

}

}

***Step 4: Compute the weighted normalized decision matrix***

*Let j = 1,2, ….n represent the criteria*

*Let i = 1,2,….s represent the alternatives*

The weights of the evaluation criteria, is multiplied by the elements of the normalized fuzzy decision matrix.

(5)

where x

The code for Step 4 is provided below.

public class Step4

{

        /// <summary>

        /// Multiplies an array of weights to an array of FuzzyNumbers

        /// </summary>

        /// <param name="normalizeArray"></param>

        /// <param name="weights"></param>

        /// <returns></returns>

public FuzzyNumber[] CalculateWeightedValues(FuzzyNumber[] normalizeArray, FuzzyNumber[] weights)

        {

            FuzzyNumber[] weightedArray = new FuzzyNumber[weights.Length];

            FuzzyNumber iFuzzyNumber;

            for (int i = 0; i < weights.Length; i++)

            {

                iFuzzyNumber = normalizeArray[i];

                FuzzyNumber weightedFuzzyNumber = new FuzzyNumber(iFuzzyNumber);

                weightedFuzzyNumber.Max = weightedFuzzyNumber.Max \* weights[i].Max;

                weightedFuzzyNumber.Mean = weightedFuzzyNumber.Mean \* weights[i].Mean;

                weightedFuzzyNumber.Min = weightedFuzzyNumber.Min \* weights[i].Min;

                weightedArray[i] = weightedFuzzyNumber;

            }

            return weightedArray;

        }

}

***Step 5: Determine the fuzzy positive-ideal solution (FPIS, A+) and the fuzzy negative-ideal solution (FNIS, A-)***

*Let j = 1,2, ….n represent the criteria*

A+ = (6)

A- = (7)

where = (1, 1, 1) and = (0, 0, 0)

The code for Step 5 is provided below.

public class Step5

{

        Utils \_utils = new Utils();

        /// <summary>

        /// Calculates the fnis.

        /// </summary>

        /// <returns>The fnis.</returns>

        /// <param name="normalizedValues">Weighted normalized Criteria values.</param>

 public FuzzyNumber CalcFNIS(FuzzyNumber[] normalizedValues)

        {

            // normalizedValues.Length is the number of Alternatives

            FuzzyNumber fnis = null;

            double[] minArray = new double[normalizedValues.Length];

            double minValue;

            // store all the min values

            for (int i = 0; i < normalizedValues.Length; i++)

            {

                minArray[i] = normalizedValues[i].Min;

            }

            // chose the lowest value

            minValue = \_utils.GetMinValue(minArray);

            fnis = new FuzzyNumber(minValue, minValue, minValue);

            return fnis;

        }

        /// <summary>

        /// Calculates the fpis.

        /// </summary>

        /// <returns>The fpis.</returns>

        /// <param name="normalizedValues">Normalized values.</param>

 public FuzzyNumber CalcFPIS(FuzzyNumber[] normalizedValues)

        {

            // normalizedValues.Length is the number of Alternatives

            FuzzyNumber fpis = null;

            double[] maxArray = new double[normalizedValues.Length];

            double maxValue;

            // store all the min values

            for (int i = 0; i < normalizedValues.Length; i++)

            {

                maxArray[i] = normalizedValues[i].Max;

            }

            // chose the lowest value

            maxValue = \_utils.GetMaxValue(maxArray);

            fpis = new FuzzyNumber(maxValue, maxValue, maxValue);

            return fpis;

        }

    }

***Step 6: Calculate the distance and of each alternative from and respectively***

*Let j = 1,2, ….n represent the criteria*

*Let i = 1,2,….s represent the alternatives*

=( (8)

=( (9)

Where d represents the distance between two triangular fuzzy numbers. For example the distance d( where =(l1,m1, u1) and =(l2,m2,u2) is expressed as follows:

d( = (10)

The code for Step 6 is provided below.

public class Step6

{

       /// <summary>

        /// Calculates the distance.

        /// </summary>

        /// <returns>The distance.</returns>

        /// <param name="fuzzyNumber">Fuzzy number.</param>

        /// <param name="idealSolution">Ideal solution. This is either the FNIS or FPIS</param>

public double CalcDistance(FuzzyNumber fuzzyNumber, FuzzyNumber idealSolution)

        {

            double distance = 0;

            double min, mean, max;

            min = Math.Pow((fuzzyNumber.Min - idealSolution.Min), 2);

            mean = Math.Pow((fuzzyNumber.Mean - idealSolution.Mean), 2);

            max = Math.Pow((fuzzyNumber.Max - idealSolution.Max), 2);

            distance = Math.Sqrt(((double)1 / 3.0) \* (min + mean + max));

            return distance;

        }

}

***Step 7: Obtain the closeness coefficient (CCi) and rank the order of alternatives***

CCi = (11)

The code for step 7 is provided below.

public class Step7

{

        /// <summary>

        /// Calculates the cci.

        /// </summary>

        /// <returns>The cci.</returns>

        /// <param name="dNegative">Fnis.</param>

        /// <param name="dPositive">Fpis.</param>

public double CalcCCI(double dNegative, double dPositive)

        {

            double cci = (dNegative / (dNegative + dPositive));

            return cci;

        }

}

The alternative with the highest closeness coefficient represents the best alternative and is closest to the FPIS and farthest from FNIS. Thereafter the ranking of the alternatives according to the closeness coefficient, CCi, in decreasing order is defined. The best alternative is closest to the FPIS and farthest to the FNIS.